PROMOTECH: A UNIVERSAL TOOL FOR PROMOTER DETECTION IN BACTERIAL GENOMES

by

© **Ruben Chevez-Guardado**

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of Master of Science.

Department of Computer Science

Memorial University

September 2020

St. John's, Newfoundland and Labrador, Canada

Abstract

A promoter is a genomic sequence where the transcription machinery binds to start copying a gene into an RNA molecule. Finding the location of bacterial promoter sequences is essential for microbiology since promoters play a central role in regulating gene expression. There are several tools to recognize promoters in bacterial genomes; however, most of them were trained on data from a single bacterium or a specific set of sigma factors. Promotech was developed to overcome this limitation, offering a machine-learning-based classifier trained to generate a model that generalizes and detects promoters in a wide range of bacterial species. During the study, two model architectures were tested, Random Forest and Recurrent Networks. The Random Forest model, trained with promoter sequences with a binary encoded representation of each nucleotide, achieved the highest performance across nine different bacteria and was able to work with short 40bp sequences and entire bacterial genomes using a sliding window. The selected model was evaluated on a validation set of four bacteria not used during training, having 50% positive and 50% negative promoter sequences resulting in an average AUPRC of 0.73±0.13 and an AUROC of 0.71±0.13. The Random Forest model achieved an average AUPRC and AUROC across the validation set's entire genomes of 0.14±0.1 and 0.71±0.17, but increased its performance to 0.75±0.18 AUPRC and 0.90±0.06 AUROC when it was configured to detect promoter clusters. Promotech was compared against state-of-the-art bacterial promoter detection programs using the balanced data set and outperformed these methods.

Acknowledgements

I would first like to thank my thesis advisor Dr. Lourdes Peña-Castillo, Ph.D. of the Department of Computer Science and Biology at Memorial University of Newfoundland and Labrador. Dr. Peña-Castillo was always willing to help and took the time to guide me in the right direction. This work would not have been possible without her help. I would also like to acknowledge Memorial University of Newfoundland and Labrador for providing financial aid that allowed me to focus on the development of this thesis. Finally, I would like to thank Compute Canada for providing the computing framework that allowed me to obtain the results necessary to complete my work.

Table of contents

List of tables

List of figures

List of symbols

σ Sigma Factor.

List of abbreviations

A	Adenine
AUPRC	Area Under the Precision Recall Curve
AUROC	Area Under the Receiver Operating Characteristics
Acc	Accuracy
BACILLUS	*Bacillus amyloliquefaciens* XH7
C	Cytosine
CC	Correlation Coefficient
CLOSTRIDIUM	*Lachnoclostridium phytofermentans* ISDg
CNN	Convolutional Neural Network
CPNEUMONIAE	*Chlamydophila pneumoniae* CWL029
CPU	Central Processing Unit
C_JEJUNI	*Campylobacter jejuni* subsp. jejuni 81116
C_JEJUNI_2	*Campylobacter jejuni* subsp. jejuni NCTC 11168
C_JEJUNI_3	*Campylobacter jejuni* RM1221
C_JEJUNI_4	*Campylobacter jejuni* subsp. jejuni 81116
C_JEJUNI_5	*Campylobacter jejuni* subsp. jejuni 81-176
DNA	Deoxyribonucleic Acid
ECOLI/ECOLI_2	*Escherichia coli* str. K-12 substr. MG1655
G	Guanine
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
GRU-X	GRU Recurrent Neural Network with X Hidden Layers
HPYLORI/HPYLORI_2	*Helicobacter pylori* 26695
ID	Identifier
K-Fold	Dataset Split into K Consecutive Folds

LINTERROGANS	*Leptospira interrogans* *serovar Manilae* isolate L495
LSTM	Long Short Term Memory
LSTM-X	LSTM Recurrent Neural Network with X Hidden Layers
MYCOBACTER	*Mycobacterium smegmatis* str. MC2 155
NGS	Next-Generation Sequencing
NN	Neural Network
RF	Random Forest
RF-HOT	Random Forest Trained with Hot-encoded Sequences
RF-TETRA	Random Forest Trained with Tetra-nucleotide Frequencies
RNA	Ribonucleic Acid
RNAP	RNA Polymerase
RODOBACTER_1/RODOBACTER_2	*Rhodobacter capsulatus* SB 1003
SCOELICOLOR	*Streptomyces coelicolor* A3(2)
SDK	Software Development Kit
SONEIDENSIS	*Shewanella oneidensis* MR-1
SPYOGENE	*Streptococcus pyogenes* strain S119
STYPHIRMURIUM	*Salmonella enterica* subsp. enterica serovar Typhimurium SL1344
SVM	Support Vector Machine
Sn	Sensitivity
Sp	Specificity
T	Thymine
TF	Transcription Factor
TSS	Transcription Start Site
bp	Base Pair
log2	Binary Logarithm
nt	Nucleotide
sqrt	Square Root

Chapter 1

Introduction

Promoters, together with transcription factors, are the first step in the process of reading and executing the instructions from the building blocks of every organism, called DNA. According to the MeSH (Medical Subject Headings) Thesaurus [2], promoters are "DNA sequence regions which are recognized (directly or indirectly) and bound by a DNA dependent RNA polymerase during the initiation of transcription and contain specific DNA sequences that are recognized by the transcription factors and the RNA polymerase." The transcription process starts when proteins called transcription factors (TF) bind to promoters to regulate which genes are expressed. TFs then recruit enzymes called RNA polymerase (RNAP), which open the double-stranded DNA helix and synthesize a single strand nucleotide sequence called messenger RNA (mRNA). mRNAs are later read by ribosomes to produce amino-acid chains called proteins that perform multiple functions within an organism. In addition to mRNAs, RNAP also synthesizes RNAs that are not translated into proteins, such as rRNAs, ncRNAs and sRNAs. Genes coding these non-coding RNAs have specialized promoters that respond to stimuli.

In this work we collected a large amount of published promoter sequence data obtained using sequencing technology such as dRNA-Seq [3] and Cappable-Seq [4] and utilized these data sets to create a machine learning model capable of generalizing and abstracting the concepts that define a promoter sequence. During the study, two machine learning model architectures are explored; the first is Random Forest (RF) [5, 6], an ensemble learning method that works by constructing multiple uncorrelated decision trees and outputting the class selected by the majority of trees. RFs are

a popular choice given their ability to reduce over-fitting. The second architecture is Recurrent Neural Network (RNN) [7], a class of neural network that can retain information through time by receiving outputs of previous time-steps as inputs and controlling the cell state using Forget/Update gates. RNNs were selected due to the sequential nature of the data and the models' ability to handle sequential prediction problems. In this study, our machine learning models receive a DNA sequence and outputs a confidence score that indicates whether or not the sequence is a promoter.

The thesis is organized as follows. Chapter 2 discusses the relevant literature review. Chapter 3 describes the data sources, how the data was processed, and the machine learning models' selection and assessment. Chapter 4 presents the results and identifies the limitations of the study, and Chapter 5 summarizes the main conclusions and suggestions for further research.

Chapter 2

Literature Review

2.1 Computational approaches for bacterial promoter recognition

There have been numerous tools developed to recognize bacterial promoter sequences. Here I will describe in chronological order these tools. Polat et al. [8] developed FS_LSSVM, which stands for feature selection and least square support vector machine. It consists of a feature dimensionality reduction of *E. coli*'s promoter sequence data set from 57 to 4 features. The second component is the least square support vector machine (FS_LSSVM) classifier. Its performance is measured by accuracy, sensitivity and specificity analysis. The *E. coli* data set consists of sequences of 57 nucleotides in length and 106 samples, including 53 promoters and 53 non-promoters. The input features are 57-nucleotide-long DNA sequences. The algorithm, without feature selection, obtained 65.38% accuracy, 70% sensitivity, 62.50% specificity using 50%-50% training-test partition and 80% accuracy using cross-validation. The model, using feature selection, was able to obtain 84.62% accuracy, 90.90%, Sensitivity, 80% Specificity using 50%-50% training-test partition and 100% accuracy using cross-validation.

Salamov et al. [9] developed BPROM, a component of the Fgenesb_annotator pipeline for promoter prediction. The pipeline identifies tRNA genes, rRNA genes, proteins, potential operons, promoters, and terminators. The pipeline's input can be a segment of a bacterial genome or short reads of DNA. BPROM uses five relatively

conserved motifs from *E. coli* to identify promoters. The conserved motifs are the -10 and 35 search area regulated by sigma 70 factor, the -60 to -40 search area upstream the -35 box with a length of 7 base pairs, the -11 to +10 search area downstream with a length of 7 base pairs, and the -31 to -22 search area with a length of 5 base pairs. Linear discriminant analysis (LDA) [10] is used to derive the recognition function from distinguishing between promoters and non-promoters sequences with a sensitivity of 83% and specificity of 84%.

Rangannan et al. [11] developed PromPredict, an algorithm that utilizes average free energy to predict promoter regions using threshold values specific to *E. coli, Bacillus subtilis*, and *Mycobacterium tuberculosis*. PromPredict was evaluated on 1144, 612, and 81 experimentally validated TSSs from *E. coli*, *B. subtilis*, and *M. tuberculosis*, respectively. It achieved a sensitivity of 99%, 95%, and 100%; and a precision of 58%, 60%, and 49% for *E. coli*, *B. subtilis*, and *M. tuberculosis*, respectively.

DeAvila Silva et al. [12] created BacPP, an algorithm "designed to recognize and predict *E. coli* promoter sequences from background with specific accuracy for each sigma factor (respectively, σ^{24}, 86.9%; σ^{28}, 92.8%; σ^{32}, 91.5%; σ^{38}, 89.3%, σ^{54}, 97.0%; and σ^{70}, 83.6%)." Sigma factors are bacterial transcription initiation factors that enables binding of RNAP to a specific gene promoter. BacPP focuses in *E. coli* and a subset of sigma factors. Its training data consists of 1,034 sequences divided by sigma factor and a set of randomly generated negative training instances. Four binary digits were used to represent each nucleotide and a Neural Network (NN) with a threshold of 0.5 is used to indicate a positive promoter detection. The optimal architecture found was a 324 input units neural network with 2-5 hidden layers (depending on the sigma factors), and one output layer for binary classification. BacPP achieved an accuracy of 76% across the sigma factor families. According to DeAvila Silva et al. [12], "In contrast to tools previously reported in the literature, BacPP is not only capable of identification of bacterial promoters in background genome sequence but is designed to provide pragmatic classification according to sigma factor."

Anne de Jong et al. [13] developed PePPER, a web-server tool to mine for regulons and Transcription Factor Binding Sites (TFBS). PePPER has a collection of Transcription Factors (TFs), TFBS and regulons of *L. lactis, E. coli* and *B. subtilis* from the RegulonDB, MolgenRegDB and DBTBS data sets publicly available via the PePPER web-server [14] . The PePPER toolbox also provides a promoter prediction

feature based on a common DNA pattern of 10 base pairs upstream of the transcription start site (TSS), a conserved sequence 35 base pairs upstream, and different sigma factors binding sites.

Solovyev et al. [15] developed CNNProm, a promoter recognition convolutional neural network (CNN) model able to detect prokaryotic and eukaryotic promoters. CNNProm is trained to detect promoters from humans, mouse, plant (*Arabidopsis*), and two bacteria (*E. coli* and *Bacillus subtilis).* The model was created using the Keras library with Theano as back-end. The network's input is based on a binary encoded representation of each nucleotide, e.g., A (1,0,0,0), T (0,1,0,0), G (0,0,1,0), C (0,0,0,1). The architecture consists of one convolutionary layer with 200 filters of a length of 21, a max-pooling layer, and a fully connected layer of 128 layers. The model achieved 0.90 sensitivity (Sn), 0.96 specificity (Sp), and 0.86 correlation coefficient (CC) on *E. coli*'s sigma 70 and Sn = 0.91, Sp = 0.95, and CC = 0.86 for *Bacillus subtilis*.

Shahmuradov et al. [16] developed bTSSfinder, "a novel tool that predicts putative promoters for five classes of sigma factors in *Cyanobacteria* (σ^A, σ^C, σ^H, σ^G and σ^F) and for five classes of sigma factors in *E. coli* (σ^{70}, σ^{38}, σ^{32}, σ^{28} and σ^{24})." bTSSfinder, similar to BacPP, focuses in *E. coli* and *Cyanobacteria'*s sigma factors, except for sigma factor 54. The data set's positive sequences consists of 251 and 1101 base pairs, divided in *E. coli* (1,544 σ^{70}, 140 σ^{38}, 237 σ^{32}, 135 σ^{28}, and 412 σ^{24}), *Nostoc* (11,386), *S. Elongatus* (1,471) and *Synechocystis* (343). The negative set consists of 8,346 *E. coli* sequences and 32,418 sequences of the three combined *Cyanobacteria* species. Additional features were included to improve accuracy such as oligomer frequencies (triplets, tetramers, pentamers and hexamers), and physico-chemical properties of DNA (free energy, base stacking, melting temperature, and entropy). bTSSfinder achieved an accuracy of 81% to 87% across the sigma factor classes.

Di Salvo et al. [1] developed G4PromFinder, "a powerful tool for promoter search in GC-rich bacteria, especially for bacteria coding for a lot of sigma factors, such as the model microorganism *S. coelicolor* A3(2)." G4PromFinder utilizes conserved motifs compared to the previous approaches which use sigma factor families. This approach was implemented using the genome sequences from *S. coelicolor* A3(2) with accession code NC_003888.3 and *P. eruginosa* PA14 with accession code NC_008463.1. The promoter identification method is based on putative promoters with maximal AT

content and G-quadruplex motifs recognition. The methods were evaluated using TSS global maps obtained by dRNA-Seq [3] experiments. G4Promfinder achieved 54% and 43% precision for *S. coelicolor* and *P. aeruginosa*, respectively.

Wang et al. [17] developed IBPP, an image-based promoter prediction method to surpass the performance of traditional promoter detection methods like the position weight matrix method (PWM) [18]. IBPP utilizes the evolutionary approach to generate images from the promoter training data, starting with a randomly generated 81 bp seed images. The images evolve from a random state, are recombined with other images, develop random mutation and filtered, leaving only the generated images with the highest similarity score to the promoter training data. The training set consists of 1,888 *E. coli* K12 MG1655 (NC_000913) promoter sequences, and the 10,000 randomly generated non-promoter sequences. IBPP is based on neural networks, and IBPP-SVM combines neural networks and SVMs. IBPP can analyze short (81bp) and long sequences (2,000bp) with a sliding window. IBPP-SVM achieved a sensitivity and specificity of 68.7% and 94.3%, respectively. IBPP achieved a sensitivity and specificity of 56.4% and 94.1%.

Oubounyt et al. [19] developed DeePromoter, a method that combines convolutional neural networks (CNN) and Long Short-term Memory RNNs (LSTM). The training data set consists of human and mouse promoter sequences, divided into TATA and non-TATA sequences. The human promoter set consists of 3,065 and 26,532 TATA and non-TATA sequences, respectively. The mouse set consists of 3,305 and 21,804 TATA and non-TATA sequences, respectively. Each positive set has its own equal size negative set. The data is represented with a binary format as follows; A, C, G, and T as (1 0 0 0), (0 1 0 0), (0 0 1 0), (0 0 0 1), respectively. The model achieves 99% precision, 99% recall, and 98% MCC, for human promoters and 99% precision 98% recall, and 97% MCC for mouse promoters.

Zhang et al. [20] developed MULTiPly, a method based on support vector machines (SVM). The training data set consists of 2,860 *E. coli K-12* positive sequences, each with a length of 81bp and the same number of negative sequences. MULTiPly can predict between promoters and non-promoters, but also classify by its sigma factor (σ^{70}, σ^{26}, σ^{32}, σ^{38}, and σ^{28}). MULTiPly achieved a sensitivity, specificity, accuracy, and MCC of (90.43, 76.93, 84.91, 0.69), (88.84, 92.91, 91.21, 0.82), (82.2, 88.41, 85.67, 0.71), (83.31, 86.68, 85.25, 0.7), and (96, 91.3, 94, 0.88), for σ^{70}, σ^{24}, σ^{32}, σ^{38} and σ^{28}

respectively.

Lai et al. [21] developed iProEP, a promoter prediction method based on support vector machines (SVM). The training data set consists of 1,787 *H. sapiens*, 1,886 *D. melanogaster*, 598 *C. elegans*, 270 *B. subtilis*, and 741 *E. coli* positive promoter sequence. The sequence lengths are 300, and 81 bp for eukaryotic and prokaryotic, respectively. iProEP achieved accuracies of 93.3%, 93.9%, 95.7%, 95.2%, and 93.1% and ROCAUCs of 0.974, 0.975, 0.981, 0.988, and 0.976 for *H. sapiens, D. melanogaster, C. elegans, B. subtilis*, and *E. coli*, respectively.

2.2 Summary

The number of promoter prediction methods is vast and worth exploring. Each method tackles the problem from different angles and targets different organisms, but all converge in a single goal, the identification of promoters. Table 2.1 was obtained from Di Salvo et al. [1] which offers a performance comparisons between G4PromFinder, PePPER, PromPredict and bTSSfinder. Table 2.2 provides an overview of some promoter identification methods published in the past twelve years.

Table 2.1: Comparison between G4PromFinder, PePPER, PromPredict and bTSSfinder. The table was obtained from Di Salvo et al. [1].

Tools	Bacterial genome	*Streptomyces coelicolor A3(2)*	*Pseudomonas aeruginosa PA14*
G4PromFinder	Recall	0.70	0.69
	Precision	0.54	0.43
	F1-score	0.61	0.53
PePPER	Recall	0.20	0.31
	Precision	0.78	0.67
	F1-score	0.32	0.42
PromPredict	Recall	0.51	0.56
	Precision	0.41	0.42
	F1-score	0.46	0.48

bTSSfinder (for *E. coli* sigma factors)	Recall Precision F1-score	0.45 0.33 0.38	0.41 0.31 0.36
bTSSfinder (for *Cyanobacteria* sigma factors)	Recall Precision F1-score	0.29 0.27 0.28	0.30 0.26 0.28

Table 2.2: Summary of promoter prediction approaches in the last twelve years. The summary includes the methods' name, publication year, the primary approach used to detect promoters, the data set used during the study, the target organism, and any particular distinguishing feature.

Name	Approach	Data set	Target Organism	Special Features
FS_LSSVM (2007)	Least square support vector machine (FS_LSSVM)	53 promoter and 53 non-promoter sequences.	*E. coli*	57 to 4 feature reduction.
BPROM (2010)	Linear Discriminant Analysis (LDA)	* The number of sequences for training are not specified.	*E. coli*	Motifs identification: -10 and 35 search area -60 to -40 search area -11 to +10 search area

PromPredict (2010)	An algorithm that uses the sequences' average free energy to differentiate between promoters and non-promoters.	1144, 612, and 81 experimentally validated TSSs from *E, coli*, *B. subtilis*, and *M. tuberculosis*, respectively.		PromPredict utilizes the average free energy sequence value for promoter identification.
BacPP (2011)	Neural Networks (NN)	1034 sequences subdivided according to their sigma factor (24, 28, 32, 38, 54, 70) and 1034 random negative sequences.	*E. coli*	BPROM can predict promoters and classify them by their sigma factor (24, 28, 32, 38, 54, 70).
PePPER (2012)	MEME motif search inside the PePPER all-in-one toolbox.	TF[1] and TFBS[2] data were downloaded from RegulonDB (E. coli) and DBTBS(B. subtilis). * The number of sequences for training are not specified.	*L. lactis*, *E. coli*, and *B. subtilis*	PePPER utilizes the Pribnow box DNA pattern for promoter identification.

[1]Transcription Factors
[2]Transcription Factor Binding Sites

CNNProm (2017)	Convolutional Neural Networks (CNN)	(P:839,N:300) *E. coli* (P:746,N:2000) *B. subtilis* (P:1426,N:8256) *Human TATA* (P:19811,N:27731) *Human non-TATA* (P:1255,N:3530) *Mouse TATA* (P:16283,N:24822) *Mouse non-TATA* (P:1497,N:2879) *Arabidopsis TATA* (P:5905,N:11459) *Arabidopsis non-TATA* Where P is for positive and N for negative promoter sequences.	CNNProm can detect promoters across selected eukaryote and prokaryote genomes.
bTSSfinder (2017)	Position weight matrices (PWM) and Neural Networks	The positive set was made of 3,597 *E. coli K12*, 12,797 cyanobacterium *Nostoc sp. PCC 7120*, 351 *Synechocystis sp. PCC 6803*, and 1,471 *S. elongatus PCC 6301* TSSs. The negative set consisted of 8,346 and 32,418 sequences for *E. coli* and the Cyanobacteria species.	bTSSfinder can classify seven different sigma classes (70, 54, 38, 32, 28, 24 and 19).
G4PromFinder (2018)	An algorithm that detects AT-rich element and G-quadruplex motifs.	The positive set consisted of 3570 *S. coelicolor A3(2)* and 2117 *P. aeruginosa PA14* sequences. The negative set consisted of 548 *S. coelicolor A3(2)* and 338 *P. aeruginosa PA14* sequences.	The prediction is based on AT-rich elements and G-quadruplex DNA motifs.

IBPP (2018)	Support Vector Machine (SVM)	1,888 positive and 10,000 negative sequences.	*E. coli K12 MG1655* σ^{70}	IBPP can analyze short (81 nt) and long (2,000 nt) sequences using a sliding window.
DeePromoter (2019)	CNNs and LSTMs	The training data set consists of human and mouse promoter sequences, divided into TATA and non-TATA sequences.	The human promoter set consists of 3,065 and 26,532 TATA and non-TATA sequences, respectively. The mouse set consists of 3,305 and 21,804 sequences, respectively, and each positive set has its own equal size negative set.	DeePromoter combines the advantages of both CNNs and LSTMs model architectures. It works with hot-encoded representations of the sequences.
MULTiPLy (2019)	Support vector machine (SVM)	2,860 promoter sequences and 2,860 non-promoter sequences.	*E. coli*	MULTiPLy is able to classify five different sigma classes (70, 38, 32, 28, 24).

iProEP (2019)	Support vector machine (SVM)	(P:1,787,N:3,600) *H. sapiens* (P:1,886,N:4,658) *D. melanogaster* (P:598,N:1,200) *C. elegans* (P:270,N:600) *B. subtilis* (P:741,N:1,400) *E. coli* Where P is for positive and N for negative promoter sequences.	iProEP can detect promoters across selected eukaryote and prokaryote genomes.

Chapter 3

Methods

This study aimed to develop a novel tool to detect promoters in bacterial genomic sequences and surpass previous methods in performance, not just in a specific bacterium or sigma factor family, but in a variety of bacterial species. To do this, we have gathered promoter sequences from many bacteria, assessed multiple iterations of the proposed machine learning models, selected and applied the correct performance metrics for the problem, and determined the optimal settings for each machine learning method. After the best models' assessment and selection, they were bench-marked against the existing tools (BPROM, G4PromFinder, bTSSFinder and MULTIPly) using a separate validation data set. Each step is described below.

3.1 Materials

3.1.1 Collecting data

Transcription Start Sites (TSS) are locations where the transcription process starts, and promoters are located upstream of the TSS. For this study, the promoters are considered to be located within 40 nucleotides (nt) upstream of the TSSs. These sequences are then pre-processed and used as input for the classification algorithms. Bacterial TSSs detected by next-generation sequencing (NGS) approaches, namely, dRNA-seq [3] and Cappable-seq [4], were collected from the literature and then transformed to promoter coordinates and FASTA sequences using BEDTools. The bacteria

used for training were *E. coli, H. pylori, C. jejuni, S. pyogenes, Salmonella enterica serovar Typhimurium, C. pneumoniae, L. interrogans*, and *S. coelicolor*. The bacteria reserved for validation were *M. smegmatis, C. phytofermentans, R. capsulatus*, and *B. amyloliquefaciens*. Each bacterium's information and literature source are described in Tables 3.1 and 3.2. The data sets consist of BED files containing a list of genomic coordinates specifying the nucleotide start and end position of TSSs in each bacterial genome. The data follows the BED format [22] and contains the following fields:

1. **Chrom:** The name of the chromosome.
2. **Start:** The starting position of the feature in the chromosome.
3. **End:** The ending position of the feature in the chromosome.
4. **Name:** An optional label to identify each feature.
5. **Score:** The score is set to zero by default in the TSS and promoter BED files, but is used to store the prediction score when running the machine learning methods.
6. **Strand:** Defines the sequence orientation as forward (+) or reverse (-).

After collecting all the TSSs, the BED files were pre-processed. The duplicated coordinates were deleted, then sorted in descending order by their genomic location, the files were moved to specific folders per bacterium, and a table referencing each file was created to be used later in the pipeline.

Table 3.1: Summary of training and validation data sets. Each bacterium is labelled with an identifier used to locate the BED and FASTA files, a second identifier (T or V) indicates if the bacterium is reserved for training or validation. Additional information is included such as the number of TSS per bacterium, the genome's length, the Next Generation Sequencing technology used to obtain the TSSs, and the literature sources' PubMed Id. The bacteria without a PubMed Id indicate that at the time of this publication, the source manuscript was still in preparation.

BACTERIUM	GENOME ACCESSION	PUBMED-ID	NGS TECH-NOLOGY	LABEL	#TSS	GENOME LENGTH	T or V
Escherichia coli str. K-12 substr. MG1655	NC_000913.3	27748404	dRNA-seq	ECOLI	278	4,699,673	T
Escherichia coli str. K-12 substr. MG1655	NC_000913.2	25266388	dRNA-seq	ECOLI_2	2,672	4,705,958	T
Helicobacter pylori 26695	NC_000915.1	20164839	dRNA-seq	HPYLORI	1,907	1,688,716	T

Helicobacter pylori 26695	NC_000915.1	30169674	dRNA-seq	HPYLORI_2	449	1,688,716	T
Campylobacter jejuni subsp. jejuni 81116	NC_009839.1	30169674	dRNA-seq	C_JEJUNI	269	1,651,375	T
Campylobacter jejuni subsp. jejuni NCTC 11168	NC_002163.1	23696746	dRNA-seq	C_JEJUNI_2	1,905	1,664,932	T
Campylobacter jejuni RM1221	NC_003912.7	23696746	dRNA-seq	C_JEJUNI_3	2,167	1,800,054	T
Campylobacter jejuni subsp. jejuni 81116	NC_009839.1	23696746	dRNA-seq	C_JEJUNI_4	1,944	1,648,467	T
Campylobacter jejuni subsp. jejuni 81-176	NC_008787.1	23696746	dRNA-seq	C_JEJUNI_5	2,003	1,720,291	T
Streptococcus pyogenes strain S119	LR031521.1	30902048	dRNA-seq	SPYOGENE	892	1,904,272	T
Salmonella enterica subsp. enterica serovar Typhimurium SL1344	NC_016810.1	22538806	dRNA-seq	STYPHIR-MURIUM	1,873	5,130,796	T
Chlamydophila pneumoniae CWL029	NC_000922.1	21989159	dRNA-seq	CPNEUMONIAE	530	1,245,608	T

Shewanella oneidensis MR-1	NC_004347.2	24987095	dRNA-seq	SONEIDENSIS	4,729	5,195,568	T
Leptospira interrogans serovar Manilae isolate L495	NZ_LT962963.1	28154810	dRNA-seq	LINTERROGANS	2,865	4,672,387	T
Streptomyces coelicolor A3(2)	NC_003888.3	27251447	dRNA-seq	SCOELICOLOR	3,570	9,168,034	T
Mycobacterium smegmatis str. MC2 155	NC_008596.1	30984135	dRNA-seq	MYCOBACTER	4,054	7,088,042	V
Lachnoclostridium phytofermentans ISDg	NC_010001.1	27982035	Cappable-seq	CLOSTRIDIUM	1,187	4,916,847	V
Rhodobacter capsulatus SB 1003	NC_014034.1	-	dRNA-seq	RODOBACTER_1	5,374	3,792,373	V
Rhodobacter capsulatus SB 1003	NC_014034.1	-	dRNA-seq	RODOBACTER_2	4,045	3,792,373	V
Bacillus amyloliquefaciens XH7	CP002927.1	26133043	dRNA-seq	BACILLUS	1,064	3,995,479	V

Table 3.2: Bacteria literature source. Each bacterium's TSS files were obtained from multiple literature sources. There are fourteen bacterial species, nine reserved for training and four for validation. A total of twenty bacterial strains were obtained for the study.

Label	Source
ECOLI	The primary transcriptome of the Escherichia coli O104:H4 pAA plasmid and novel insights into its virulence gene expression and regulation [23]
ECOLI_2	Global transcriptional start site mapping using differential RNA sequencing reveals novel antisense RNAs in Escherichia coli [24]
HPYLORI	The primary transcriptome of the major human pathogen Helicobacter pylori [25]
HPYLORI_2	ANNOgesic: a Swiss army knife for the RNA-seq based annotation of bacterial/archaeal genomes [26]
C_JEJUNI	ANNOgesic: a Swiss army knife for the RNA-seq based annotation of bacterial/archaeal genomes [26]
C_JEJUNI_2	High-resolution transcriptome maps reveal strain-specific regulatory features of multiple Campylobacter jejuni isolates [27]
C_JEJUNI_3	High-resolution transcriptome maps reveal strain-specific regulatory features of multiple Campylobacter jejuni isolates [27]
C_JEJUNI_4	High-resolution transcriptome maps reveal strain-specific regulatory features of multiple Campylobacter jejuni isolates [27]
C_JEJUNI_5	High-resolution transcriptome maps reveal strain-specific regulatory features of multiple Campylobacter jejuni isolates [27]
SPYOGENE	Conserved and specific features of Streptococcus pyogenes and Streptococcus agalactiae transcriptional landscapes [28]
STYPHIRMURIUM	The transcriptional landscape and small RNAs of Salmonella enterica serovar Typhimurium [29]
CPNEUMONIAE	The transcriptional landscape of Chlamydia pneumoniae [30]
SONEIDENSIS	Conservation of transcription start sites within genes across a bacterial genus [31]

LINTERROGANS	Genome-wide transcriptional start site mapping and sRNA identification in the pathogen Leptospira interrogans [32]
SCOELICOLOR	The dynamic transcriptional and translational landscape of the model antibiotic producer Streptomyces coelicolor A3(2) [33]
MYCOBACTER	Defining the transcriptional and post-transcriptional landscapes of Mycobacterium smegmatis in aerobic growth and hypoxia [34]
CLOSTRIDIUM	Global repositioning of transcription start sites in a plant-fermenting bacterium [35]
RODOBACTER_1	Gruell et al, manuscript in preparation
RODOBACTER_2	Gruell et al, manuscript in preparation
BACILLUS	The Global Transcriptional Landscape of Bacillus Amyloliquefaciens XH7 and High-Throughput Screening of Strong Promoters Based on RNA-seq Data [36]

3.1.2 Generating positive set and negative set

After filtering and sorting the TSS files, a pipeline was designed to obtain the positive and negative promoter sets. It was created using Nextflow 20.04.1 build 5335, Java SDK 1.8.0_252, Python 3.6, BEDTools v2.29.2, and Docker v19.03.6. Nextflow [37] is used for its ability to separate the desired functionality in components and use them in parallel or sequentially, and it uses BEDTools [38] through genomicpariscentre/bedtools [39], a Docker container used to trigger BEDTools without additional software installations. The BEDTools [38] library provides multiple components to transform FASTA sequences and BED coordinates files. It is used in the pipeline to convert the TSS to promoter coordinates, and the promoter coordinates to FASTA sequences. The FASTA sequences are then pre-processed and used as inputs for the machine learning methods.

The first step in the pipeline is to generate a genome length file per bacterium automatically. The Nextflow pipeline takes a genome FASTA file as input, calculates the length, and saves it on a ".genome" file. BEDTools' `slopBed` [38] command is then used to generate the positive set of promoter coordinates. It takes the genome and the TSS BED files as input and generates the promoter coordinates with a length of 40 nt each. The parameters used to execute the command are (-i) indicates the

TSS BED file input, (-g) the genome file input, (-r) indicates how many base pairs are added to the right, and (-l) how many base pairs added to the left. For example,

```
slopBed -i  ECOLI_TSS.bed -s -g ECOLI.genome
-l 39 -r 0 > positive.bed
```

The next step is to generate the FASTA sequences from the promoter coordinates. To do it, the BEDTools' `getFasta` [38] command is used. The parameters used to execute the command are (-bed) the promoter coordinates file as input, (-fi) the bacteria's genomic sequence as input, (-s) forces strandedness, and (-bed) the output promoter FASTA sequences.

```
bedtools getfasta -s -bed positive.bed
-fi ECOLI.fasta -fo positive.fasta
```

The negative coordinates are generated randomly with a length of 40 base pairs using BEDTools' `random` [38] command. The parameters are (-l) The length of each coordinate and (-n) the total number of coordinates in the BED file.

```
bedtools random -l 40 -n ‘grep -i -v ‘‘>" positive.fasta | wc -l’ -g
ECOLI.genome > tmp_negative_promoter.bed
```

In real-life, the total number of promoters in a bacterium's genome is minimal compared to the total number of 40bp segments in the whole genome. Due to this, the pipeline was modified to generate an unbalanced negative set of 1:10 ratio, as shown in the command below.

```
bedtools random -l 40 -n ‘grep -i -v ‘‘>" positive.fasta | echo
$(($(wc - l) * 10 ))’ -g ECOLI.genome > tmp_negative_promoter.bed
```

To avoid duplicated coordinates appearing in both positive and negative sets, the `subtract` [38] command is used to delete them from the negative set if they have a minimum sequence overlap of 13% with the sequences in the positive set (-f). The other parameters are (-s) force strandedness, (-A) removes the entire coordinate if it exceeds the threshold, (-a) the randomly generated set of coordinates, and (b) the positive coordinates set file path.

```
bedtools subtract -f 0.13 -s -A -a tmp_negative_promoter.bed
-b positive.bed > negative.bed
```

The command `getfasta` [38] is used again to obtain the FASTA sequences from the negative coordinates file, and this concludes all the steps in the pipeline for a single bacterium. The pipeline is orchestrated by a python file that runs it for every bacterium and feeds the pipeline the required parameters and can be set up to output a balanced or 1:10 ratio unbalanced data set. The pipeline's output is 55,146 sequences from all bacteria combined, having 50% positives and 50% negatives when it is set up to output a balanced data set and 301,006 when set up to output an unbalanced set of 10% positives and 90% negatives. The training-testing split is done after the sequences are pre-processed.

Table 3.3: Training data sets.

ID	**BALANCED**		**UNBALANCED**	
BACTERIA ID	POSITIVE	NEGATIVE	POSITIVE	NEGATIVE
ECOLI	248	278	248	2,773
ECOLI_2	2,636	2,616	2,636	26,147
HPYLORI	1,877	1,829	1,877	18,273
HPYLORI_2	448	445	448	4,448
CJEJUNI	269	268	269	2,674
CJEJUNI_2	1,881	1,843	1,881	18,298
CJEJUNI_3	2,140	2,069	2,140	20,736
CJEJUNI_4	1,919	1,870	1,919	18,619
CJEJUNI_5	1,973	1,927	1,973	19,173
SPYOGENE	891	880	891	8,754
STYPHIR-MURIUM	1,869	1,845	1,869	18,464
CPNEUMO-NIAE	530	520	530	5,221
SONEIDEN-SIS	4,728	4,661	4,728	46,511

LINTERRO-GANS	2,791	2,803	2,791	27,979
SCOELI-COLOR	3,566	3,526	3,566	35,170
SUBTOTAL	27,766	27,380	27,766	273,240
TOTAL	55,146		301,006	

3.2 Machine learning models

During the study, two types of machine learning methods were used, Recurrent Neural Networks (RNNs) [7] and Random Forest (RF) [5, 6]. Both methods have been successfully used before to classify genomic sequences. Random Forest is a popular machine learning method for its ability to identify feature importance, handles many data types (continuous, categorical and binary), is well-suited for high-dimensional data, and avoids over-fitting by its voting-scheme among the ensemble of trees within it [40]. According to Zhang et al. [40], RF models have been used in micro-array gene expression analysis due to the high-dimensional and high ratio of noise in the data. Lee et al. [41], compared K-nearest Neighbors (KNN) [42], Linear Discriminant Analysis variants (LDA) [43], bagging trees [44], boosting [45], RFs, and concluded that RF was the most effective technique used in the micro-array data sets. Izmirlian et al. [46] and Kirchner et al. [47], demonstrated that RFs are well-suited for mass spectrometry-based proteomic data analysis, known for its high noise ratio and high dimensional data. Segal et al. [48] and Hamby et al. [49] have demonstrated the use of RFs for biological sequence analysis. Segal et al. [48] have developed a model capable of predicting the replication capacity of virus based on amino acid sequences and Hamby et al. [49] have developed a model for glycosylation sites prediction.

Recurrent Neural Networks are also well-suited for genomic sequence analysis due to their ability to handle variable-length inputs, detecting sequential patterns, and retaining information through time. Hill et al. [50] have utilized RNNs for detecting protein-coding potential trained on human messenger RNA (mRNA) and long non-coding RNA (lncRNA) sequences. Also, Shen et al. [51] used RNNs for predicting transcription factor binding sites (TFBS). Considering all the advantages of both

types of models and their variations, RFs and RNNs were selected for this study. The RF models were trained using two types of inputs, hot-encoded binary representation vectors and tetra-nucleotide frequencies. The RNN models were trained using word embeddings [52].

3.2.1 Data pre-processing

Hot-encoded features

A Random Forest [5, 6] model was trained using hot-encoded features; this meant that the nucleotides (A, G, C, T) were transformed into binary vector representations [1000], [0100], [0010], and [0001] respectively. Multiple nested loops are used to go through each bacterium, each promoter sequence, and a sliding window of one nt size and one nt step converting each nucleotide to its binary representation. Each promoter sequence is 40 base pairs long with each nucleotide having a four-digit representation resulting in 160 features.

Tetra-nucleotide frequencies

A second Random Forest [5, 6] model was trained using tetra-nucleotide frequencies calculated using the scikit-bio library [39]. Tetra-nucleotide frequencies were used as training features because the Random Forest models have a useful functionality of creating feature importance rankings that allowed us to find the most important tetra-nucleotide sequences during training based on their frequencies' importance scores, as discussed in detail in Section 4.2. To obtain the tetra-nucleotide frequencies, each promoter sequence is fed to the "kmer_frequencies" method [53], which returns a dictionary of tetra-nucleotides with their frequencies. An empty 2D table is created with 256 columns. The columns represent all the possible arrangements of tetra-nucleotides or four nucleotide sequences, i.e. AAAA, AAAT, AATT, etc. Each tetra-nucleotide and its frequency is then mapped to its column position in the table and the tetra-nucleotide combinations not present in the sequence are left with a value of zero. The output number of features is 256 possible tetra-nucleotide frequencies.

Word embeddings

The Recurrent Neural Networks [7] were trained using word embeddings or a numerical representation of the tetra-nucleotide sequences. Multiple nested loops were used to go through each bacterium, each promoter sequence, and a sliding window of four nt size and one nt step to obtain 37 tetra-nucleotides per promoter sequence. The 37 tetra-nucleotides are then joined together into a single string separated by spaces and then fed to the Keras' Tokenizer class [54] . The Tokenizer class is used for text processing and requires a sentence-like input, then it takes every word and obtains a numerical representation for every unique word in the data set. The output number of features is 37 tokens representing each tetra-nucleotide in the promoter sequence.

3.2.2 Model architectures and implementation

Random forest with hot encoding

The model is created using the Sklearn's RandomForestClassifier [54] combined with the GridSearchCV [55] python class and expects the hot-encoded data set as input. The GridSearchCV class handles the best hyper-parameters' search. It does this by creating multiple models with different hyper-parameters combinations, assessing them, and retaining only the best model. The hyper-parameters used during the search are "max_features": [None, "sqrt", "log2"] and "n_estimators": [1000, 2000, 3000]. The remaining parameters were left as default. The default parameters were the following:

```
    criterion=gini, max_depth=None, min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True,
oob_score=False, n_jobs=None, random_state=None, verbose=0,
warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None
```

The "max_features" (m) parameter indicates the subset of features considered during a node split. If m is None, then m is equal to the total number of features, "sqrt" is the square root of the number of features, and "log2" is the binary logarithm of the number of features. The "n_estimators" (n) parameter indicates the number of trees trained in the ensemble. The model was trained using an imbalanced data with a 1:10

ratio between positive and negative instances. The best selected hyper-parameters were m="log2" and n=2,000. The class weights were automatically generated with a value of class 0 = 0.53 and class 1 = 10.28.

Random forest with tetra-nucleotides

Random forest is used again for a second model, but this one expects the tetra-nucleotide frequencies data set as input. The architecture, parameters, and class weights are the same as described above and the hyper-parameters selected by the grid search are m="log2" and n=2,000. The main differences between the models are the input shape, 160 features for hot encoding and 256 for tetra-nucleotides frequencies.

Recurrent Neural Networks

Two types of RNNs were tested during the study, Long Short-Term Memory Unit (LSTM) [56] and Gated Recurrent Unit (GRU) [57]. Both models try to solve the vanishing and exploding gradient problems that occur in the vanilla RNN architecture. GRU uses two gates instead of LSTM's three gates and are considered less complex and more memory efficient. Originally, the models were built around the Scikit-learn's GridSearchCV [55] for hyper-parameter optimization and the KerasClassifier [54] wrapper to handle Keras models from within Scikit-learn's methods. But, each grid-search iteration proved to be very time-consuming and computationally expensive. It was decided to create separate models with different number of layers, trained and tuned manually. The models were made using Keras [54] and Tensorflow v2.1.0 [58] as back-end. The only hyper-parameter that varied was the number of hidden layers. The remaining parameters are described in Table 3.4. The models' architecture consisted of one embedding layer [54] with 50 neurons and a vocabulary size automatically calculated by the Tokenizer [54] class in the pre-processing pipeline. Also, one GRU or LSTM layer, a sigmoid activation, a dropout layer, and a binary output. Each layer had 100 neurons.

Table 3.4: Recurrent Neural Networks' hyper-parameters.

Hyper-parameters	Values

Embedding layer neurons	50
RNN layer neurons	100
Hidden layer neurons	100
# Hidden layers	[0, 1, 2, 3, 4]
Dropout rate	0.2
Epochs	50
Batch size	10
Vocabulary size	301
Activation function	Sigmoid
Optimizer function	Adam
Loss function	Binary cross-entropy

3.3 Model assessment

The RF and RNN models were trained on the Compute Canada's Beluga Cluster [59] configured with four NVidia V100SXM2 16GB GPUs, eight Intel Gold 6148 Skylake @ 2.4 GHz CPUs, and managed through SLURM commands. The first step was to select the best performing architecture for each type of model. The 1:10 unbalanced bacteria set reserved for training/testing were used to train the models. First, it was pre-processed using the Nextflow pipeline and joined together in a single set. The data's total size was 496,041 sequences with 160 features for the hot-encoding set, 256 features for the tetra-nucleotide frequencies set, and 37 features for the word embedding set. The data set reserved for training consisted of nine bacteria and was split in 90% training, 10% testing. Four bacteria were reserved for validation. The two RF models were trained using GridSearchCV with a K-Fold value of 10. The RNNs were tuned and trained manually with a total of 10 combinations using GRU and LSTM layers, each with 0, 1, 2, 3 or 4 hidden layers. The grid-search for both RF models selected m="log2", and n=2,000 as the best hyper-parameters. The RNN models and weights were stored using the Keras "save_model" [54] method and the RF models were serialized into binary files using the Joblib [60] python library.

3.4 Benchmarking

The best models obtained during training and testing were compared to the previous methods BProm [9] , bTSSfinder [16] , G4Promfinder [1], and MULTiPLy [20]. Three tests were performed, the first consisted in scanning a whole bacterial genome using a sliding window and predicting the probability of finding a promoter in each window. The second test used the same approach but instead of trying to locate a specific promoter location, it focused in finding clusters where the probability of finding a promoter was high. The third test used a balanced data set with the same number of positive and negative samples. It is important to highlight that it was not possible to evaluate the four other tools in test one and two because they were not designed to be run in a whole bacterial genome and their running time would have been unfeasible.

3.4.1 Test 1: Whole genome promoter prediction

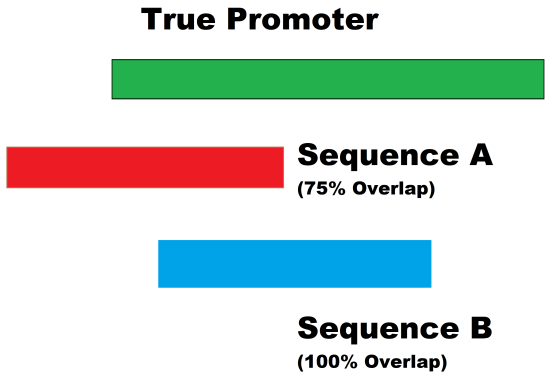
A sliding window of 40 nt size with a step of one nt was used to scan the whole genome. The sliding window travelled along the genome, appending each sequence to a vector that later was used as the pipeline's input to transform the sequences to the correct format expected by each model. The number of generated sequences ranged from 4 to 7 million depending on the genome size. After obtaining the list of sequences, it was fed to the pipeline, which transformed the data into a hot-encoding and word embedding format.

The pipeline was unable to generate the whole genome's tetra-nucleotide frequencies due to a repeated python "Memory Error", which indicated that the device was running out of RAM memory. The code was optimized to use Numpy [61] arrays and methods. Vector concatenations were avoided to reduce complexity and memory allocation. Despite the code and memory management optimization, the large quantity of data combined with the use of scikit-bio's tetra-frequency calculation still maxed the available memory. Only the RF model trained with hot encoded features, and the RNNs were able to be used during tests one and two.

After completing the data transformation, the data sets were fed to the RNNs and the RF model. The predictions were obtained in a BED file containing the coordinates for each position scanned and the score. The results were then assessed using

BEDTools by comparing the predictions coordinates, with the coordinates 40bp upstream of the TSS detected by the NGS technologies. The BEDTools' **intersect** [38] command was used to obtain the number of correctly predicted promoter sites. The command filtered all the predictions that had at least 0.1 overlap with the true promoter sequences, as shown in Figure 3.1.

Figure 3.1: The sequence overlap represents the percentage of nucleotides in the predicted promoter sequence that are located in the same genomic region as the true promoter sequence. All the predicted sequences that have a 10% overlap or more are considered corrected predicted promoter sequences. Sequence A and Sequence B in the figure represent predicted promoters.



The process was repeated multiple times, the score threshold was changed, the predictions were filtered, and only those with scores above the threshold (0.5, 0.6, 0.7, 0.8, and 0.9) were left. This was done to check which score threshold gave the best recall versus precision performance. The confusion matrix per score threshold was calculated, and used to draw the Precision-Recall and ROC curves.

3.4.2 Test 2: Whole genome cluster prediction

As described in detail in Section 4.3, after visual inspection of the predicted sequences compared to the true promoters, we observed that predicted promoter sequences grouped or clustered near the true promoter sequences. Thus, we designed a second test were sequences not necessary overlapping but very close to the true promoters were considered correct. This test was almost identical to the first, the only difference

was that instead of using **intersect** [38], it was the **closest** [38] command. The closest command filtered all the coordinates adjacent in distance to the true promoter. It was configured to filter all but the five prediction closest to the real promoters. This parameter is called "the closest hits" (k) and had a value of five. After being filtered by BEDTools, a second filter was applied, which only left those coordinates less than 100 nt in distance in both directions from the true promoters. The cluster prediction was done to find nearby predictions that did not align perfectly with the real promoters but were located in their proximity. The process was repeated per score threshold (0.5, 0.6, 0.7, 0.8, and 0.9), the confusion matrices were calculated, and the performance graphs were generated.

3.4.3 Test 3: Balanced data set prediction

The third test included the trained models during the study and four state-of-the-art promoter prediction programs (BPROM [9], BTSSFinder [16], G4PromFinder [1], and MULTiPLy [20]). The TSS coordinates from the bacteria reserved for validation were fed to the pipeline to obtain the transformed representations for the RNN and RF models. The generated data set contained the same number of positive and negative sequences, as shown in Table 3.5, and each sequence had a length of 40 nt. The RFs, RNNs, G4PromFinder, and MULTiPLy accepted 40 nt inputs, but BProm and BTSSFinder required longer sequences of 250 nt. The BEDTools' **slopBed** [38] command was used to extend the existing sequences in both directions from 40 to 250 nt. G4Promfinder was easier to integrate to the pipeline since it was developed in python and the sequences were fed directly from the script. BProm and bTSSfinder required writing each sequence to disc and then running the programs through a shell script from the python pipeline and this significantly increased the execution time. Finally, MULTiPLy was tested separately, since it was developed in Matlab, so a short script was created to feed and evaluate each bacterium's data set.

Table 3.5: Balanced test data set.

BACTERIA ID	PROMOTER SEQUENCES	NON-PROMOTER SEQUENCES
MYCOBACTER	4,054	3,978
CLOSTRIDIUM	1,187	1,177
RHODOBACTER_1	5,374	5,207
RHODOBACTER_2	4,045	3,918
BACILLUS	1,064	1,055
SUBTOTAL	15,724	15,335
TOTAL	31,059	

3.5 Summary

A set of TSS coordinates were collected from a wide range of bacteria obtained from different literature sources. The TSSs were then fed together with the bacterial genomes to the pre-processing pipeline to obtain the promoter coordinates and sequences. The pipeline used the BEDTools commands to obtain the promoter coordinates from the TSS coordinates and the promoter sequences from the promoter coordinates. It generated random coordinates and sequences used as the negative data set. Two types of models were used during the study, Random Forest (RF) and Neural Networks (RNNs). The RFs were sub-divided into two types, one using promoter sequences transformed to hot-encoded representation vectors and the other using promoter sequences transformed to tetra-nucleotide frequencies. RNNs were sub-divided into LSTMs and GRUs, which were then manually tuned with 0, 1, 2, 3, and 4 hidden layers. After training the models, they were compared with methods from previous studies. Three tests were made during the benchmark; the first and second tests used a sliding window that travelled through a whole bacterial genome and produced a BED file with coordinates and scores. The predictions were then assessed using BEDTools' intersect command to filter only the predictions that correctly overlapped the true promoters at least a certain threshold. The second test was similar, but instead of using intersect, it used the closest command, which filtered and left only the prediction coordinates closest to the true promoters. The third test utilized a balanced data set of bacteria reserved for validation. The positive and negative sets contained the same number of sequences. In the third test, the positive set was made of the true promoters from each bacterium, and the negative set was generated from random sequences coming from each bacterial genome.

Chapter 4

Results and Discussion

4.1 Model assessment

A total of two Random Forest [5, 6] models and ten variants of Recurrent Neural Networks [7] were trained during this study. As described in Section 3.2.1, the training data for the RF models was obtained by pre-processing the promoter sequences to extract their tetra-nucleotide frequency and hot-encoding representations. The RNN models consisted of five Long Short-Term Memory Unit (LSTM) [56] and five Gated Recurrent Unit (GRU) [57] models having zero to four hidden layers and a word embedding [52] representation of the promoter sequences as input. The balanced data set consisted of nine bacteria reserved for training and testing with a total of 55,146 sequences, as shown in Table 3.3. The data set was split in 75% training and 25% testing. The two RF models obtained the best Area Under the Precision Recall Curve (AUPRC) and Area Under the Receiver Operating Characteristic Curve (AUROC) during training and testing, as shown in Table 4.1.

Table 4.1: Training and testing AUPRC and AUROC performance results using a balanced data set.

	TRAINING		TESTING	
MODELS	AUPRC	AUROC	AUPRC	AUROC
RF-HOT	0.99	0.99	0.95	0.94
RF-TETRA	0.98	0.98	0.86	0.86

GRU-0	0.82	0.81	0.81	0.80
GRU-1	0.57	0.57	0.58	0.57
GRU-2	0.52	0.52	0.52	0.52
GRU-3	0.55	0.55	0.55	0.56
GRU-4	0.50	0.49	0.51	0.50
LSTM-0	0.91	0.90	0.91	0.90
LSTM-1	0.91	0.91	0.91	0.90
LSTM-2	0.91	0.90	0.91	0.90
LSTM-3	0.57	0.54	0.54	0.56
LSTM-4	0.50	0.50	0.50	0.50

The total number of promoters in a bacterial genome is much smaller than the total number of 40 nt segments in the whole genome. For that reason, the models trained with a balanced data set were tested on a 1:10 ratio unbalanced set to determine if they could be suitable for whole-genome promoter recognition. The original data set was expanded by increasing the number of random negative promoter sequences ten times, ending in an unbalanced set of 301,006 sequences, as shown in Table 3.3. The data set was split in 90% training and 10% testing. The models were tested again using the unbalanced testing set and obtained a lower performance compared to the previous balanced test, as shown in Tables 4.1 and 4.2.

Table 4.2: AUPRC and AUROC performance results using a 1:10 ratio unbalanced test data set.

MODELS	AUPRC	AUROC
RF-HOT	0.70	0.74
RF-TETRA	0.43	0.40
GRU-0	0.51	0.51
GRU-1	0.51	0.50
GRU-2	0.51	0.51
GRU-3	0.46	0.46
GRU-4	0.75	0.50
LSTM-0	0.50	0.51
LSTM-1	0.50	0.50

LSTM-2	0.49	0.49
LSTM-3	0.50	0.60
LSTM-4	0.75	0.50

The low performance of the models with the unbalanced sets suggested that the models would not be suitable for real life scenarios, and hence, 12 new models were trained using a 1:10 ratio unbalanced data set to meet the requirements needed for use on an entire genome. The RF model trained with hot-encoded sequences obtained the highest performance, followed by the GRU with one layer and LSTM with three layer, as shown in Table 4.3. It is important to note that the RNNs considered during the study have a set architecture with only one variable hyper-parameter, being the number of hidden layers. Due to the long training time constraints, no more variations were considered. It took around 20 hours to train each RNN model on the balanced data compared to the 12 hours taken by the automatically tuned RF models using grid-search hyper-parameter optimization. The RNNs took 6.5 days to train on the unbalanced data, and the models had a file size of 1 to 3 MB; meanwhile, the RFs took four days of training and had a file size of 3.3 to 3.7GB. In summary, with enough time and resources, there is still room for optimization in the RNNs' architecture and the RF models take less time to train but have a higher footprint in disk. The four best models RF-HOT, RF-TETRA, GRU-1, and LSTM-4 were selected for the final tests using *Mycobacterium smegmatis, Lachnoclostridium phytofermentans, Rhodobacter capsulatus*, and *Bacillus amyloliquefaciens'* complete genomes reserved for validation (i.e., not seen during training).

Table 4.3: Training and testing AUPRC and AUROC performance results from the twelve new models using the 1:10 ratio unbalanced data set.

	TRAINING		TESTING	
MODELS	AUPRC	AUROC	AUPRC	AUROC
RF-HOT	0.990	0.999	0.802	0.938
RF-TETRA	0.961	0.997	0.593	0.844
GRU-0	0.761	0.932	0.752	0.929
GRU-1	0.788	0.939	0.778	0.934

GRU-2	0.764	0.932	0.753	0.929
GRU-3	0.732	0.921	0.728	0.922
GRU-4	0.740	0.923	0.728	0.923
LSTM-0	0.739	0.924	0.734	0.923
LSTM-1	0.747	0.927	0.744	0.927
LSTM-2	0.742	0.924	0.739	0.923
LSTM-3	0.749	0.926	0.748	0.924
LSTM-4	0.757	0.929	0.748	0.928

4.2 Feature analysis

Before proceeding with the validation tests, a feature analysis was done to find possible patterns and motifs recognized by the models. The RF classifiers can create a feature importance ranking based on importance scores. The scores were calculated using Permutation-based Importance or Mean Decrease Accuracy (MDA) [62] and Impurity-based Importance [6]. First, the RF model trained with tetra-nucleotide frequencies was used to calculate the average impurity-based feature importance across the ensemble of trees for each tetra-nucleotide sequence. The most important sequences based on their frequencies were TATA, ATAA, TAAT, TTAT, AAAA, and TTTT, which can be considered as conserved motifs across the wide range of bacteria used during the study. The test was repeated but using the permutation-based importance score and a permutating each feature five times. Both tests produced similar results having the same tetra-nucleotide sequences appearing at the top of the ranking, only varying their ranking position and score, as shown in Tables 4.4 and 4.5. The results indicate that the RF models focus on a motif called Pribnow-Schaller box [63, 64], which is a six nucleotide consensus sequence (TATAAT), commonly located around ten base pairs upstream from the transcription start site.

Table 4.4: Impurity-based feature importance ranking generated using the Random Forest model trained with tetra-nucleotide frequencies.

IMPURITY-BASED FEATURE IMPORTANCE

IMPORTANCE RANKING	TETRA-NUCLEOTIDE SEQUENCE	IMPORTANCE SCORE
1	TATA	0.023±0.015
2	ATAA	0.014±0.009
3	TAAT	0.014±0.008
4	TTAT	0.011±0.007
5	AAAA	0.010±0.001
6	TTTT	0.010±0.001
7	GTTA	0.009±0.004
8	TATT	0.009±0.004
9	TAAA	0.009±0.002
10	AATA	0.008±0.004
11	TTTA	0.008±0.003
12	AATT	0.008±0.003
13	AAAT	0.008±0.002
14	ATTT	0.008±0.002
15	ATTA	0.008±0.004
16	GCTA	0.008±0.003
17	TTAA	0.007±0.002
18	CAGC	0.007±0.004
19	GTAT	0.007±0.003
20	CAAA	0.007±0.001

Table 4.5: Permutation-based feature importance ranking generated using the Random Forest model trained with tetra-nucleotide frequencies.

PERMUTATION-BASED FEATURE IMPORTANCE		
IMPORTANCE RANKING	TETRA-NUCLEOTIDE SEQUENCE	IMPORTANCE SCORE
1	ATAA	0.052±0.001
2	TATA	0.048±0.001
3	TAAT	0.046±0.002

4	TTAT	0.039±0.001
5	GTTA	0.036±0.001
6	TAAA	0.035±0.001
7	AATA	0.035±0.001
8	ATTA	0.033±0.001
9	TATT	0.031±0.001
10	AATT	0.031±0.000
11	AAAT	0.030±0.001
12	GCTA	0.029±0.001
13	TTTA	0.028±0.001
14	GATA	0.027±0.001
15	ATTT	0.026±0.000
16	CAAA	0.026±0.001
17	AAAA	0.024±0.001
18	GGTA	0.023±0.001
19	GTAT	0.023±0.000
20	CCAA	0.021±0.001

The feature importance analysis was repeated with the RF model trained with hot-encoded features. To give some context about the data's format, each of the 40 nucleotides was transformed into a four-digit binary representation, expanding the feature-set from 40 to 160 features. Each feature represents the existence of having one of the four possible nucleotides adenine (A), thymine (T), guanine (G), and cytosine (C) for the current position in the range of -39 to 0 relative to the transcription start site (TSS). The features were expanded using a 4-digit binary representation per nucleotide, e.g. A (1000), G (0100), C (0010), and T (0001). The permutation and impurity-based feature importance ranking generated by the RF model provided the most important positions in the range of -39 to 0 relative to the TSS and the nucleotide with the most occurrence for each position. The results demonstrated the large concentration of adenine (A) and thymine (T) in the range of -8 to -12 relative to the TSS as shown in Tables 4.6 and 4.7. Each nucleotide's importance score was plotted on a bar graph to have visual representations of the results. Figures 4.1, and 4.2 show an AT-rich concentration with higher importance values on the positions -10

and -35 following the Pribnow-Schaller box pattern [63, 64].

Table 4.6: Impurity-based feature importance ranking generated using the Random Forest model trained with hot encoded promoter sequences.

IMPURITY-BASED FEATURE IMPORTANCE			
IMPORTANCE RANKING	NUCLEOTIDE	POSITION	IMPORTANCE SCORE
1	A	-11	0.043±0.021
2	T	-7	0.040±0.020
3	T	-12	0.023±0.010
4	T	-8	0.019±0.007
5	A	-9	0.019±0.012
6	A	-12	0.019±0.007
7	A	-10	0.018±0.008
8	C	-11	0.017±0.011
9	G	-11	0.014±0.008
10	T	-11	0.014±0.006
11	G	-8	0.014±0.009
12	T	-13	0.013±0.004
13	A	-8	0.013±0.006
14	A	0	0.013±0.008
15	T	-6	0.013±0.005
16	T	-0	0.012±0.005
17	G	-7	0.012±0.007
18	C	-12	0.012±0.006
19	A	-1	0.012±0.005
20	G	-14	0.011±0.006

Figure 4.1: Impurity-based feature importance scores per nucleotide per position relative to the transcription start site.

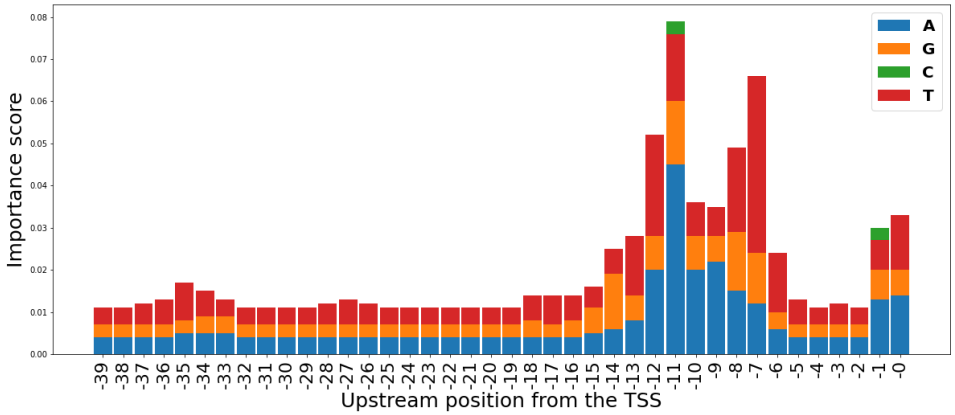
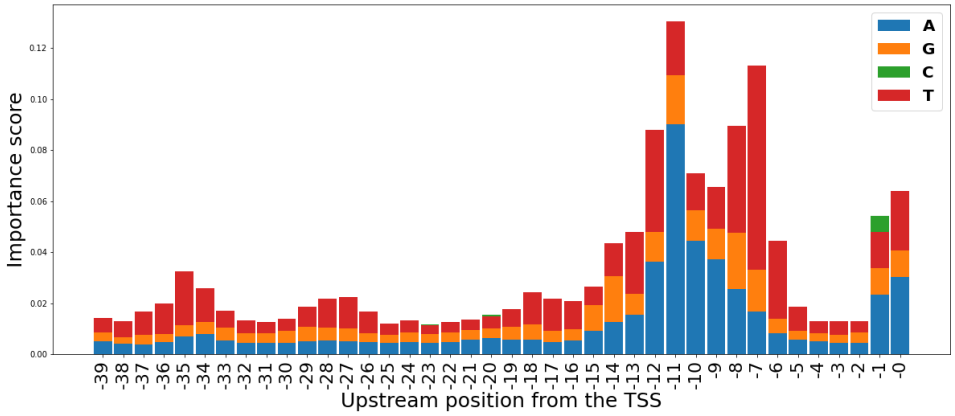


Table 4.7: Permutation-based feature importance ranking generated using the Random Forest model trained with hot encoded promoter sequences.

PERMUTATION-BASED FEATURE IMPORTANCE			
IMPORTANCE RANKING	NUCLEOTIDE	POSITION	IMPORTANCE SCORE
1	A	-11	0.090±0.002
2	T	-7	0.080±0.002
3	A	-10	0.044±0.001
4	T	-8	0.042±0.001
5	T	-12	0.040±0.001
6	A	-9	0.037±0.001
7	A	-12	0.036±0.001
8	T	-6	0.031±0.001
9	A	0	0.030±0.001
10	A	-8	0.026±0.001
11	T	-13	0.024±0.001
12	A	-1	0.023±0.001
13	T	0	0.023±0.001
14	G	-8	0.022±0.001

15	C	-11	0.021±0.001
16	T	-35	0.021±0.001
17	T	-11	0.021±0.001
18	C	-1	0.020±0.001
19	A	-12	0.019±0.001
20	G	-11	0.019±0.001

Figure 4.2: Permutation-based feature importance scores per nucleotide per position relative to the transcription start site.



4.3 Benchmarking

After completing the feature analysis, the preparation of the validation tests started. The goal was to create a pipeline able to receive small genomic sequences, but to also traverse whole bacterial genomes. This proved to be a very demanding task for the models since they were trained to receive a 40 nt input, and it was necessary to build a sliding window with a one nt step and a 40 nt window size. The sliced sequences were then pre-processed and fed to the model twice. First, using a forward strand configuration and then using a backwards strand configuration. These same steps were repeated for *Mycobacterium smegmatis, Lachnoclostridium phytofermentans, Bacillus amyloliquefaciens* and *Rhodobacter capsulatus* reserved for validation. For example, the sliding window created 6,988,167 sequences of 40 nt when used on

the *Mycobacterium smegmatis'* genome. Each sequence was repeated a second time with a backward strand configuration ending up with 13,976,336 sequences and each of the four selected models was executed around 14 million times for each of the four validation bacterial species. The process took around 4 hours to run per bacterium, including the sliding window, the data pre-processing, and the model's execution. Increasing the step size decreased the time during execution but also decreased the model's performance. The complete genomes' predicted true promoters were filtered using the BEDTools **intersect** [38] command, with confidence thresholds of 0.5, 0.6, 0.7, 0.8, 0.9, and overlap threshold of 0.1. The AUPRC and AUROC were calculated based on the results as shown in Tables 4.8 and 4.9.

Table 4.8: AUPRC performance per model and per bacterium using the BEDTools' intersect command to count the predicted true promoters. The numbers in bold indicate the model with the highest performance.

BACTERIA	RF-HOT	GRU-0	GRU-1	LSTM-3	LSTM-4
MYCOBACTER	**0.271**	0.016	0.014	0.035	0.039
CLOSTRIDIUM	0.019	0.005	0.005	**0.024**	0.02
RHODOBACTER_1	**0.119**	0.040	0.031	0.058	0.062
RHODOBACTER_2	**0.106**	0.032	0.026	0.05	0.056
BACILLUS	**0.189**	0.015	0.010	0.019	0.011

Table 4.9: AUROC performance per model and per bacterium using the BEDTools' intersect command to count the predicted true promoters. The numbers in bold indicate the model with the highest performance.

BACTERIA	RF-HOT	GRU-0	GRU-1	LSTM-3	LSTM-4
MYCOBACTER	0.646	**0.568**	0.58	0.696	0.666
CLOSTRIDIUM	**0.834**	0.546	0.562	0.66	0.61
RHODOBACTER_1	0.558	0.57	0.567	**0.691**	0.663
RHODOBACTER_2	0.563	0.572	0.575	**0.699**	0.668

BACILLUS	**0.924**	0.593	0.599	0.66	0.585

Table 4.10: Average AUPRC and AUROC performance plus–minus standard deviation per model across the bacteria validation set using the BEDTools' intersect command to count the predicted true promoters. The numbers in bold indicate the model with the highest performance.

MODELS	AVERAGE AUPRC ACROSS BACTERIA	AVERAGE ROC-AUC ACROSS BACTERIA	RANKING BASED ON AUPRC	THRESHOLD FOR BEST RECALL	THRESHOLD FOR BEST PRECISION
RF-HOT	**0.141±0.095**	**0.705±0.166**	**1**	**0.5**	**0.6**
GRU-0	0.028±0.015	0.697±0.060	4	0.5	0.6
GRU-1	0.028±0.016	0.683±0.045	4	0.5	0.6
LSTM-3	0.037±0.017	0.681±0.022	3	0.5	0.6
LSTM-4	0.038±0.022	0.638±0.038	2	0.5	0.6

The AUPRC seems low, but if one considers that the test was extremely unbalanced, having millions of 40 nt-long genomic sequences considered as negatives and a couple of thousand sequences considered as positives, the performance is much better than random. For example, for *M. smegmatis* there are four thousand true promoters and 14 million genomic sequences, thus a random classifier would have an AUPRC of 0.0003 which compared with the AUPRC obtained by Random Forest model using hot-encoded sequences (RF-HOT) is a thousand-fold improvement over random performance. The RF-HOT showed the best overall AUPRC and AUROC performance across the bacteria, as shown in Table 4.10. After visual inspection of the predicted promoters, it was concluded that many predictions were located adjacent to the real promoter sequences but did not overlap with them, as shown in Figure 4.3. To account for this, a second approach was considered using the BEDTools' **closest** [38] command to count as correct predictions those adjacent to the real promoter. The

closest command was configured to select the five nearest predictions to the true promoters. The nearest predictions were filtered again to select the predictions within 100 nt in both directions. We called this task "the cluster promoter prediction". Assessing the performance of the models using the cluster promoter prediction method increased AUPRC 2 to 6 times and AUROC by 1 to 1.5 times the values obtained with the "intersect" test, as shown in Tables 4.11 and 4.12.

Figure 4.3: Promoter prediction clusters discovered in the true promoters' proximity but not overlapping. Blue squares on the first row indicate the location of true promoters while blue squares on the second and third rows indicate the location of predicted promoters with a score of 0.6 and 0.5 respectively.

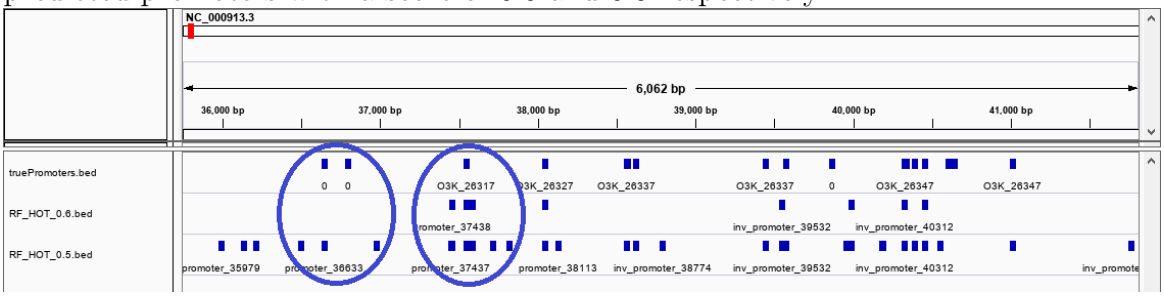


Table 4.11: AUPRC performance per model and per bacterium using the BEDTools' closest command to count the predicted true promoter clusters. The numbers in bold indicate the model with the highest performance.

BACTERIA	RF-HOT	GRU-0	GRU-1	LSTM-3	LSTM-4
MYCOBACTER	**0.868**	0.43	0.49	0.415	0.376
CLOSTRIDIUM	**0.428**	0.273	0.311	0.269	0.264
RHODOBACTER_1	**0.844**	0.736	0.715	0.681	0.661
RHODOBACTER_2	**0.84**	0.672	0.688	0.626	0.6
BACILLUS	**0.787**	0.286	0.327	0.25	0.214

Table 4.12: AUROC performance per model and per bacterium using the BEDTools' closest command to count the predicted true promoter clusters. The numbers in bold indicate the model with the highest performance.

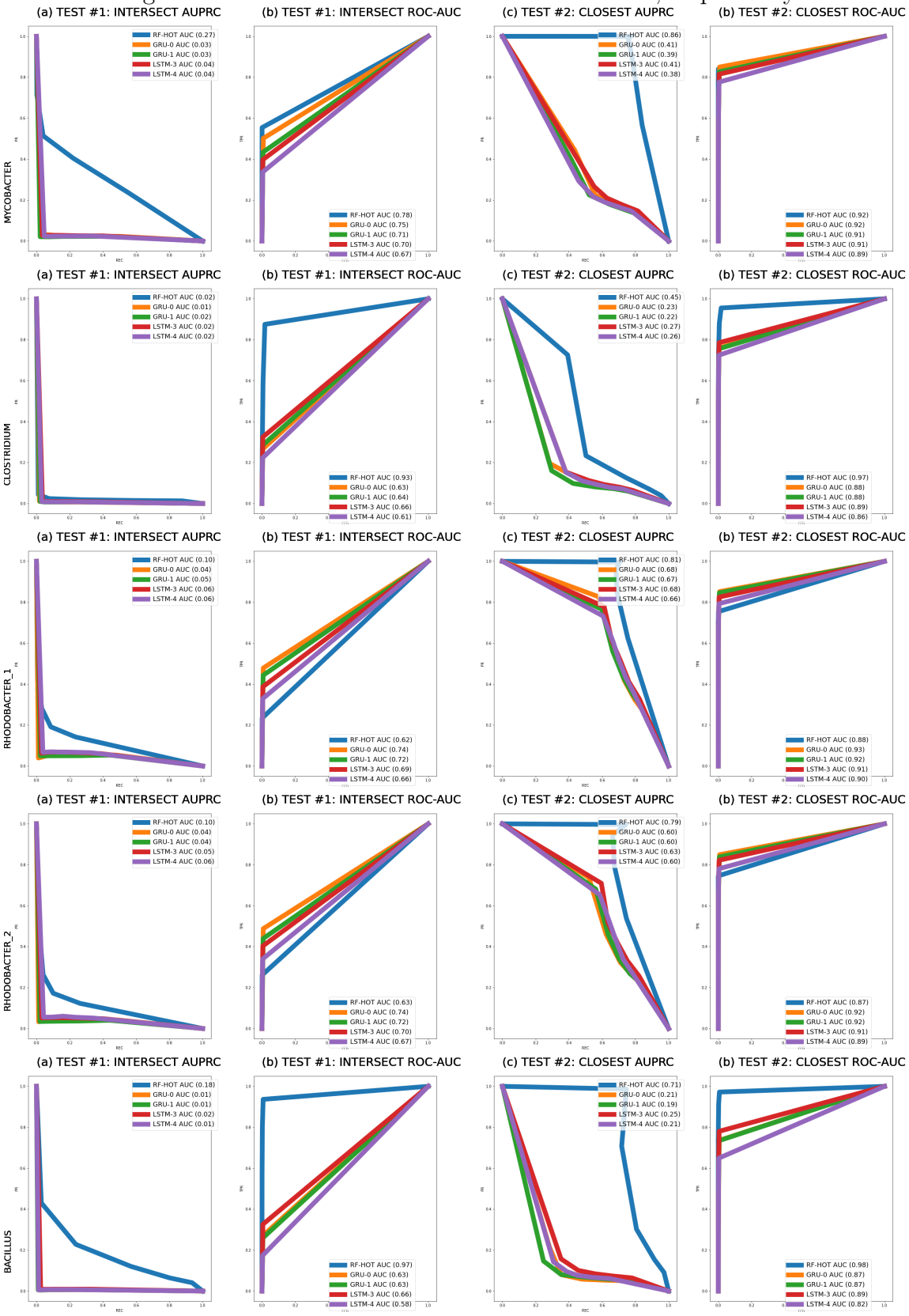
BACTERIA	RF-HOT	GRU-0	GRU-1	LSTM-3	LSTM-4
MYCOBACTER	**0.895**	0.805	0.829	0.906	0.887
CLOSTRIDIUM	**0.95**	0.799	0.831	0.892	0.861
RHODOBACTER_1	0.856	0.85	0.859	**0.912**	0.896
RHODOBACTER_2	0.843	0.833	0.851	**0.91**	0.889
BACILLUS	**0.968**	0.783	0.8	0.889	0.823

The average AUPRC, AUROC scores and thresholds across the bacteria were calculated using the performance results from test two and are shown in Table 4.13. The RF model using hot-encoded sequences showed the best overall performance across the bacteria. Figure 4.4 illustrates the improvement gained using cluster prediction compared to the previous test using the BEDTools' intersect command. This suggests that our models predict promoters in the proximity of actual promoters but these predicted promoters do not match the exact genomic location of the actual promoters.

Table 4.13: Average AUPRC and AUROC performance per model across the bacteria validation set using the BEDTools' closest command to filter the true predicted promoter clusters. The numbers in bold indicate the model with the highest performance.

MODELS	AVERAGE AUPRC ACROSS BACTERIA	AVERAGE AUROC ACROSS BACTERIA	RANKING BASED ON AUPRC	THRESHOLD FOR BEST RECALL	THRESHOLD FOR BEST PRECISION
RF-HOT	**0.753±0.184**	**0.902±0.055**	**1**	**0.5**	**0.6**
GRU-0	0.427±0.212	0.903±0.028	3	0.5	0.6
GRU-1	0.411±0.216	0.899±0.026	5	0.5	0.6
LSTM-3	0.448±0.199	0.902±0.011	2	0.5	0.6
LSTM-4	0.423±0.199	0.871±0.030	4	0.5	0.6

Figure 4.4: Comparison between the AUPRC and AUROC obtained in test one and test two using the BEDTools intersect and closest command, respectively.



The third test was made using a more conventional approach using 40 nt-long genomic sequences as input. As in the previous tests, the validation data set included *Mycobacterium smegmatis, Lachnoclostridium phytofermentans, Bacillus amyloliquefaciens* and *Rhodobacter capsulatus* with a 1:1 ratio of positive to negative instances. The smaller data set allowed the inclusion of methods from previous studies and the RF model trained with tetra-nucleotide frequencies that failed to run on a whole genome as previously explained in Section 3.4.1. The benchmark results in terms of AUPRC and AUROC are shown in Table 4.14, Table 4.15, and Figure 4.5. The RF model trained with hot-encoded genomic sequences had the best overall AUPRC and AUROC across the bacteria.

Table 4.14: AUPRC performance per model and per bacterium using a balanced validation set. The numbers in bold indicate the model with the highest performance.

BACTERIA	RF-HOT	RF-TETRA	GRU-0	GRU-1	LSTM-3	LSTM-4
MYCOBACTER	**0.951**	0.8	0.646	0.622	0.625	0.623
CLOSTRIDIUM	**0.631**	0.608	0.486	0.49	0.499	0.501
RHODOBACTER_1	**0.704**	0.678	0.588	0.576	0.559	0.573
RHODOBACTER_2	**0.732**	0.697	0.561	0.537	0.531	0.527
BACILLUS	0.639	**0.843**	0.486	0.5	0.494	0.505

Table 4.15: AUROC performance per model and per bacterium using a balanced validation set. The numbers in bold indicate the model with the highest performance.

BACTERIA	RF-HOT	RF-TETRA	GRU-0	GRU-1	LSTM-3	LSTM-4
MYCOBACTER	**0.934**	0.814	0.63	0.601	0.622	0.592
CLOSTRIDIUM	0.597	**0.608**	0.488	0.487	0.489	0.47
RHODOBACTER_1	0.665	**0.674**	0.577	0.566	0.546	0.593
RHODOBACTER_2	0.692	**0.694**	0.549	0.527	0.51	0.595
BACILLUS	0.677	**0.837**	0.496	0.502	0.481	0.506

Figure 4.5: Comparison between the AUPRC and AUROC performance obtained in test three using a balanced validation set.

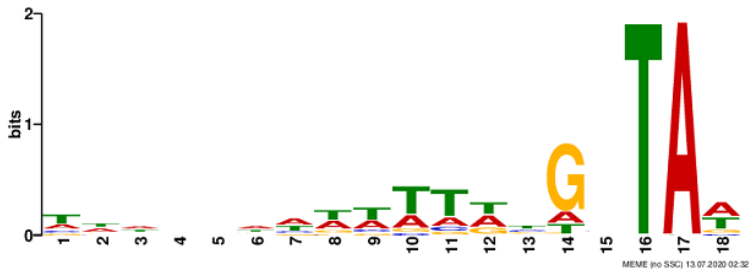
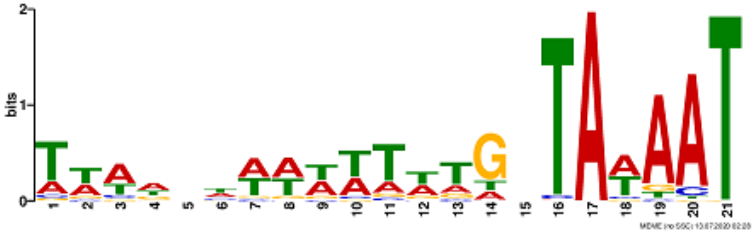
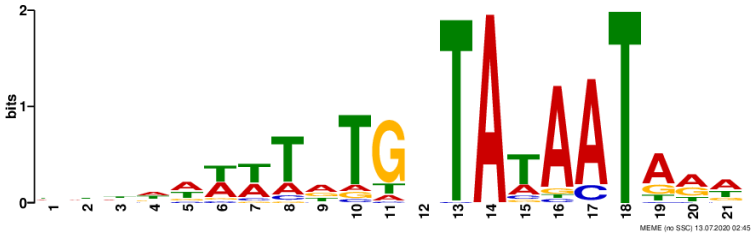
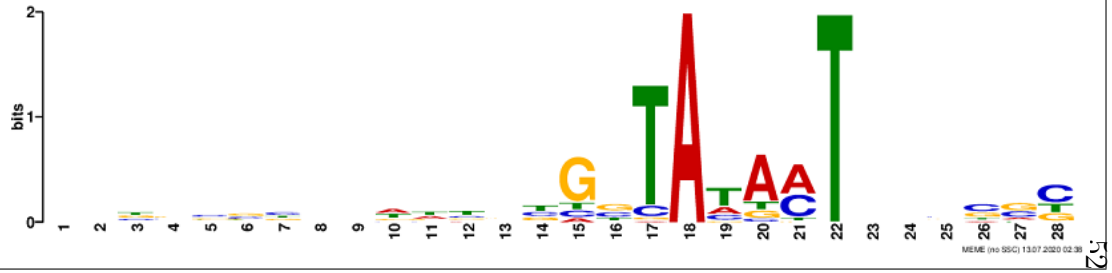
Table 4.16: Average AUPRC and AUROC performance ± standard deviation per model across the five bacterial strains in a balanced validation set. The numbers in bold indicate the model with the highest performance.

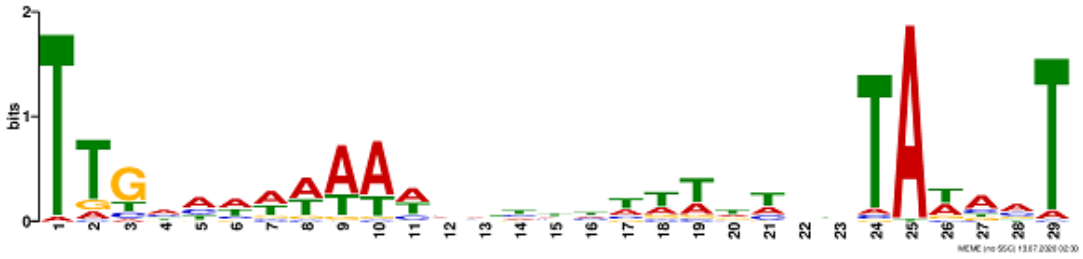
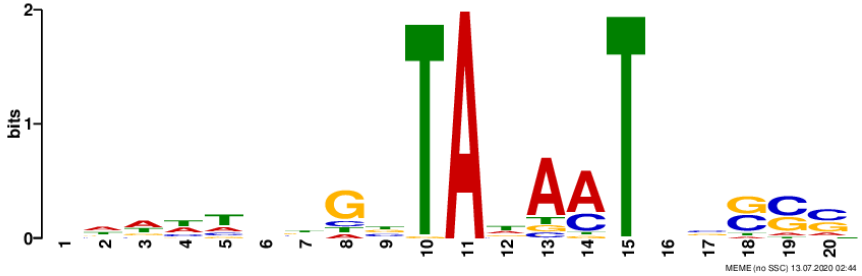
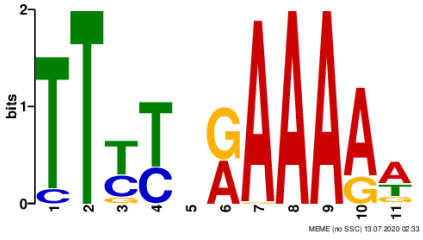
MODELS	AVERAGE PRECISION AT BEST THRESHOLD	AVERAGE RECALL AT BEST THRESHOLD	AVERAGE AUPRC	AVERAGE AUROC	RANKING BASED ON PRECISION AT BEST THRESHOLD	RANKING BASED ON AUPRC	RANKING BASED ON AUROC
RF-HOT	**0.603±0.176**	**0.946±0.061**	**0.731±0.130**	**0.713±0.129**	**3**	**1**	**2**
RF-TETRA	**0.606±0.097**	**0.902±0.070**	**0.725±0.095**	**0.725±0.097**	**2**	**2**	**1**
GRU-0	0.505±0.003	1.0±0.0	0.553±0.069	0.548±0.059	5	4	4
GRU-1	0.505±0.003	1.0±0.0	0.545±0.055	0.537±0.047	5	4	4
LSTM-3	0.505±0.003	1.0±0.0	0.542±0.054	0.530±0.057	5	4	4
LSTM-4	0.505±0.003	1.0±0.0	0.546±0.052	0.551±0.059	5	4	4
MULTiPly	0.530±0.039	0.809±0.135	0.592±0.072	0.608±0.092	4	3	3
G4PROMFINDER	0.471±0.056	0.683±0.372	-	-	6	-	-
BTSSFINDER	0.409±0.228	0.345±0.353	-	-	7	-	-
BPROM	0.682±0.093	0.191±0.248	-	-	1	-	-

RF-HOT had the best performance in terms of AUPRC on all the validation data sets but the *Bacillus amyloliquefaciens*' where RF-TETRA had the best performance. Both RF models had the best performance across the bacteria as shown in Figure 4.5. A deeper analysis was made to find additional properties found in each bacterium to help explain the models' possible target patterns. Table 4.17 shows a large concentration of proteo-bacteria with high AT content in the training data set. Table 4.18 shows that there is only one proteo-bacteria and lower AT content in the validation data set. These motifs were obtained with a computational program and have not been validated. The Multiple Em for Motif Elicitation (MEME) [65] tool was used to find conserved motifs in the bacterial promoter sequences. Table 4.18 shows that *Lachnoclostridium phytofermentans* and *Bacillus amyloliquefaciens* had very noticeable TATAAT conserved motifs compared to the other bacteria. Additionally, *Mycobacterium smegmatis* had a high concentration of thymine (T) and guanine (G) in the -10 position relative to the TSS. Both nucleotides proved to have higher importance score in that position, as shown in Figures 4.1 and 4.2.

Table 4.17: Training data set's characteristics. Note that the motif's X-axis is labelled from 1 to 40 but the the positions relative to the transcription start site are -39 to 0.

TRAINING BACTERIA	TYPE	MEDIAN GC %	MEME MOTIFS
E. coli	Proteobacteria	50.6	

Helicobacter pylori	Proteobacteria	38.9	
Campylobacter jejuni	Proteobacteria	30.4	
Streptococcus pyogenes	Firmicutes	38.4	
Salmonella enterica serovar Typhimurium	Proteobacteria	52.1	

Chlamydia pneumoniae	Chlamydiae	40.6	
Shewanella oneidensis	Proteobacteria	46	
Leptospira interrogans	Spirochaetes	35	

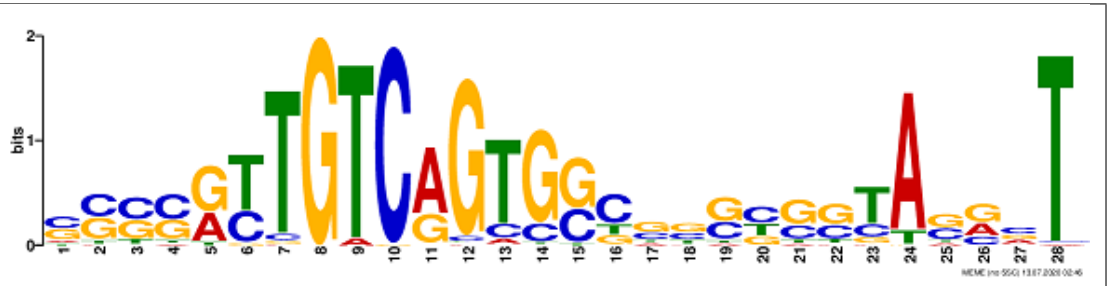
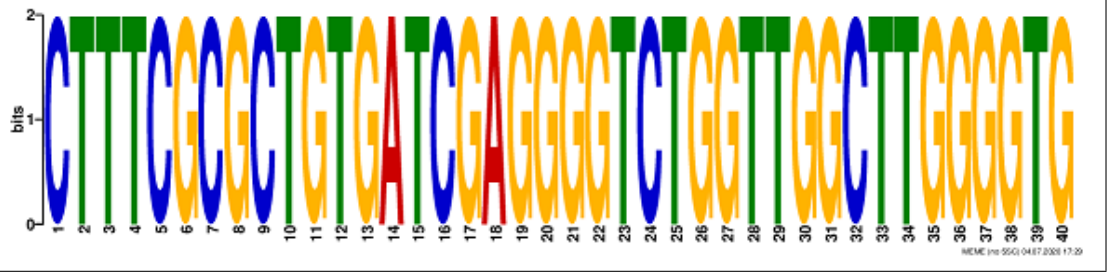
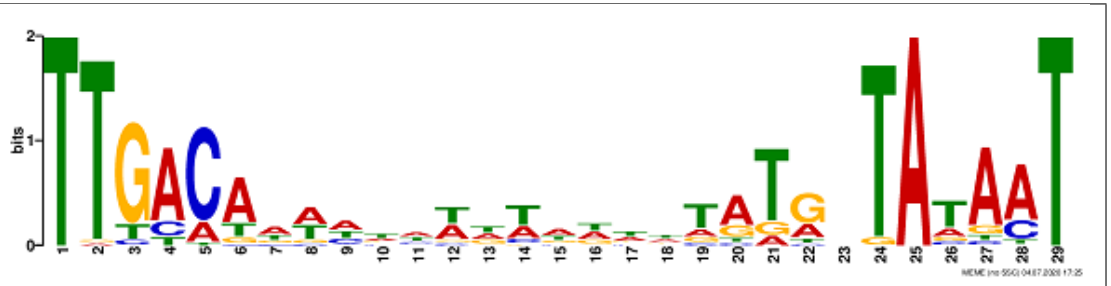
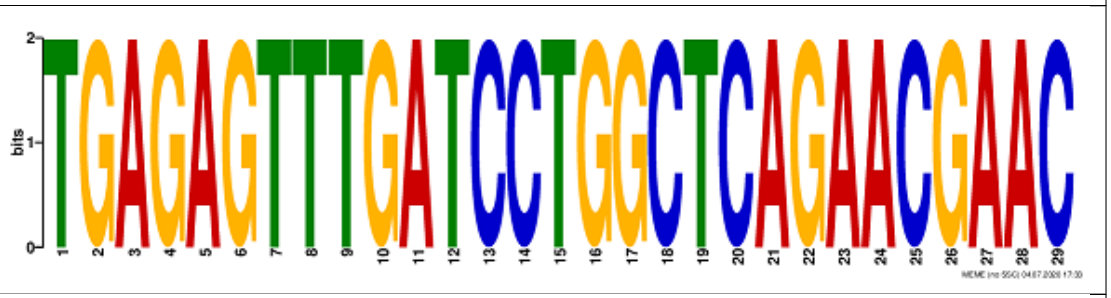
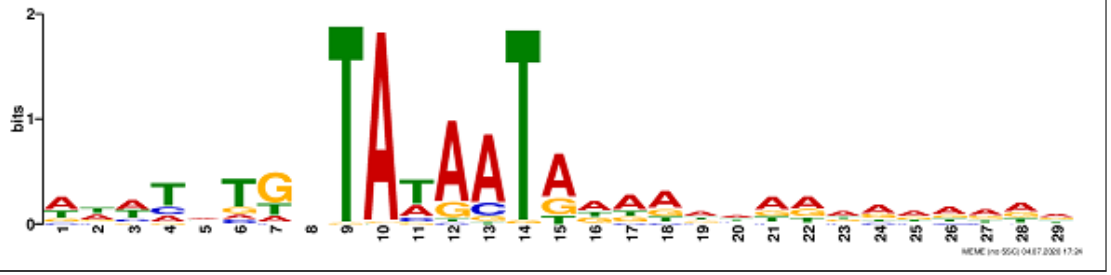
Streptomyces coelicolor	Actinobacteria	71.98	

Table 4.18: Validation data set's characteristics. Note that the motif's X-axis is labelled from 1 to 40 but the the positions relative to the transcription start site are -39 to 0.

VALIDATION BACTERIA	TYPE	MEDIAN GC %	MEME MOTIFS
Mycobacterium smegmatis	Actinobacteria	67.4	

Lachnoclostridium phytofermentans	Firmicutes	35.6	
Rhodobacter capsulatus	Proteobacteria	66.5	
Bacillus amyloliquefaciens	Firmicutes	46.4	

4.4 Summary

The Random Forest trained with hot-encoded promoter sequences was selected as the Promotech's default predictive model for whole genome assessment and both RFs were selected for balanced data sets, due to their high performance across the bacteria validation tests. The performance on the first whole genome test, where the model was tasked to predict the promoters' exact position, was RF-HOT (0.14±0.1 AUPRC and 0.71±0.17 AUROC). During the second test, the model was tasked to predict promoter clusters and increased its performance to RF-HOT (0.75±0.18 AUPRC and 0.9±0.06 AUROC). The last test used a balanced data set, including four state-of-the-art promoter prediction programs (BPROM, BTSSFinder, G4PromFinder, and MULTiPLy) and, the RF models outperformed all the other programs across all the bacteria with a performance of RF-HOT (0.73±0.13 AUPRC and 0.71±0.13 AUROC) and RF-TETRA (0.73±0.1 AUPRC and 0.73±0.1 AUROC).

Chapter 5

Conclusions

In this project, we built a universal prokaryotic promoter recognition model, and, based on the complete empirical analysis of promoter patterns found across a wide range of bacteria and the results obtained by the Random Forest and Recurrent Neural Network models, we concluded the following:

1. Both Random Forest models' feature importance ranking showed that AT-rich sequences had the highest importance scores, as shown in Tables 4.4, 4.5, 4.6, and 4.7. These sequences matched the Pribnow-Schaller box consensus sequence, which is a commonly found in bacterial promoter regions.

2. The Random Forest models gave higher importance scores to the Adenine (A) and Thymine (T) nucleotides located around the -10 and -35 positions relative to the TSS, as shown in Figures 4.1 and 4.2. The pattern is more noticeable in some bacteria than others, for example, in *L. phytofermentans* and *B. amyloliquefaciens* is more present compared to *M. smegmatis* and *R. capsulatus*, as shown in Table 4.18.

3. The Random Forest model trained with hot-encoding sequences outperformed the Recurrent Neural Networks by around 20-40% AUPRC and 1-4% AUROC, as shown in Tables 4.10, 4.13, and 4.16. Additionally, Random Forest took 40% less time to train, as shown in Section 4.1. Note that there is room for improvement in the RNNs' architecture, as the RNNs' hyper-parameters were manually tuned due to their long training time of 6.5 days using the unbalanced

data set. Tuning them using automated processes, such as grid-search, proved to be challenging due to the exponential time increase with every new possible hyper-parameter combination included in the search. In sum, there might be different hyper-parameter combinations that might increase their performance.

4. The Random Forest model trained with hot-encoded sequences and the Recurrent Neural Networks were able to traverse entire genomes using a 40 nt sliding window with long running times of around four hours and 14 million executions per genome.

5. The Random Forest model trained with tetra-nucleotide frequencies was unable to traverse the entire genome due to a higher demand for random-access memory (RAM) allocation space and CPU intensive tasks needed to obtain the tetra-nucleotide frequencies from, around, 14 million sliding window sequences per bacterium.

6. Since the RF models predicted promoters nearby actual promoters, their average AUPRC performance substantially increased when predictions within 100 nt of an actual promoter were considered correct, as shown in Tables 4.10, 4.13 and Figure 4.4. For example, the RF-HOT model increased its performance from 0.141±0.095 to 0.753±0.184 average AUPRC. This indicates that the models detect signals in the region close to an actual promoter; although they may miss the exact location.

7. The RNN models achieved a high average AUPRC and AUROC performance during training and testing but performed poorly on unseen bacteria in the validation tests, indicating that the models over-fitted the training data. For example, the GRU-0 model's performance dropped from 0.761 during training to 0.553±0.069 average AUPRC during the balanced validation test, as shown in Tables 4.3 and 4.16.

8. The Random Forest models proved to produce an overall high average AUPRC and AUROC performance across a wide range of bacteria in different conditions such as balanced, unbalanced, and whole genomes' data sets, as shown in Tables 4.10, 4.13, and 4.16. Both models outperformed bTSSfinder and G4Promfinder on average precision at the best thresholds by around 30% and MULTiPLy's AUPRC by around 15% on the balanced validation test.

Based on these findings, it can be stated that Promotech is a well-suited tool for universal bacterial promoter detection when compared to currently available tools that have a more narrow focus on a single bacterium or small set of bacteria and sigma factors. Promotech is available at GitHub [66].

Bibliography

[1] Marco Di Salvo, Eva Pinatel, Adelfia Talà, Marco Fondi, Clelia Peano, and Pietro Alifano. G4PromFinder: an algorithm for predicting transcription promoters in GC-rich bacterial genomes based on AT-rich elements and G-quadruplex motifs. *BMC bioinformatics*, 19(1):36, 2018.

[2] Darlene Chapman. Advanced search features of pubmed. *Journal of the Canadian Academy of Child and Adolescent Psychiatry*, 18(1):58, 2009.

[3] Cynthia M Sharma and Jörg Vogel. Differential RNA-seq: the approach behind and the biological insight gained. *Current Opinion in Microbiology*, 19:97–105, 2014.

[4] Laurence Ettwiller, John Buswell, Erbay Yigit, and Ira Schildkraut. A novel enrichment strategy reveals unprecedented number of novel transcription start sites at single base resolution in a model prokaryote and the gut microbiome. *BMC genomics*, 17(1):199, 2016.

[5] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282, 1995.

[6] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[8] Kemal Polat and Salih Günecs. A novel approach to estimation of *E. coli* promoter gene sequences: Combining feature selection and least square support vector machine (FS_LSSVM). *Applied mathematics and computation*, 190(2):1574–1582, 2007.

[9] Victor Solovyevand Asaf Salamov and A Solovyevand. Automatic annotation of microbial genomes and metagenomic sequences. *Metagenomics and its applications in agriculture. Nova Science Publishers, Hauppauge*, pages 61–78, 2011.

[10] Patricia Cohen. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Routledge, 2014.

[11] Vetriselvi Rangannan and Manju Bansal. Relative stability of DNA as a generic criterion for promoter prediction: whole genome annotation of microbial genomes with varying nucleotide base composition. *Molecular bioSystems*, 5(12):1758–1769, 2009.

[12] Scheila de Avila e Silva, Sergio Echeverrigaray, and Günther J L Gerhardt. BacPP: bacterial promoter prediction — a tool for accurate sigma-factor specific assignment in enterobacteria. *Journal of theoretical biology*, 287:92–99, 2011.

[13] Anne de Jong, Hilco Pietersma, Martijn Cordes, Oscar P Kuipers, and Jan Kok. PePPER: a webserver for prediction of prokaryote promoter elements and regulons. *BMC genomics*, 13(1):299, 2012.

[14] Anne de Jong and Others. PePPER Web Server. `http://pepper.molgenrug.nl/`, 2012.

[15] Ramzan Kh Umarov and Victor V Solovyev. Recognition of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks. *PloS one*, 12(2), 2017.

[16] Ilham Ayub Shahmuradov, Rozaimi Mohamad Razali, Salim Bougouffa, Aleksandar Radovanovic, and Vladimir B Bajic. bTSSfinder: a novel tool for the prediction of promoters in *Cyanobacteria* and *Escherichia coli*. *Bioinformatics*, 33(3):334–340, 2017.

[17] Sheng Wang, Xuesong Cheng, Yajun Li, Min Wu, and Yuhua Zhao. Image-based promoter prediction: a promoter prediction method based on evolutionarily generated patterns. *Scientific reports*, 8(1):1–9, 2018.

[18] Gary D Stormo, Thomas D Schneider, Larry Gold, and Andrzej Ehrenfeucht. Use of the 'Perceptron'algorithm to distinguish translational initiation sites in *E. coli*. *Nucleic acids research*, 10(9):2997–3011, 1982.

[19] Mhaned Oubounyt, Zakaria Louadi, Hilal Tayara, and Kil To Chong. DeePromoter: Robust promoter predictor using deep learning. *Frontiers in genetics*, 10:286, 2019.

[20] Meng Zhang, Fuyi Li, Tatiana T Marquez-Lago, André Leier, Cunshuo Fan, Chee Keong Kwoh, Kuo-Chen Chou, Jiangning Song, and Cangzhi Jia. MULTiPly: a novel multi-layer predictor for discovering general and specific types of promoters. *Bioinformatics*, 35(17):2957–2965, 2019.

[21] Hong-Yan Lai, Zhao-Yue Zhang, Zhen-Dong Su, Wei Su, Hui Ding, Wei Chen, and Hao Lin. iProEP: a computational predictor for predicting promoter. *Molecular Therapy-Nucleic Acids*, 17:337–346, 2019.

[22] UCSC. BED Format. `https://genome.ucsc.edu/FAQ/FAQformat.html`, 2000.

[23] Petya Berger, Michael Knödler, Konrad Förstner, Michael Berger, Christian Bertling, Cynthia M Sharma, Jörg Vogel, Helge Karch, Ulrich Dobrindt, and Alexander Mellmann. The primary transcriptome of the *Escherichia coli* O104: H4 pAA plasmid and novel insights into its virulence gene expression and regulation. *Scientific reports*, 6:35307, 2016.

[24] Maureen K Thomason, Thorsten Bischler, Sara K Eisenbart, Konrad Förstner, Aixia Zhang, Alexander Herbig, Kay Nieselt, Cynthia M Sharma, and Gisela Storz. Global transcriptional start site mapping using differential RNA sequencing reveals novel antisense RNAs in *Escherichia coli*. *Journal of bacteriology*, 197(1):18–28, 2015.

[25] Cynthia M Sharma, Steve Hoffmann, Fabien Darfeuille, Jérémy Reignier, Sven Findeißand Alexandra Sittka, Sandrine Chabas, Kristin Reiche, Jörg Hackermüller, and Richard Reinhardt. The primary transcriptome of the major human pathogen *Helicobacter pylori*. *Nature*, 464(7286):250–255, 2010.

[26] Sung-Huan Yu, Jörg Vogel, and Konrad U Förstner. ANNOgesic: a Swiss army knife for the RNA-seq based annotation of bacterial/archaeal genomes. *GigaScience*, 7(9):giy096, 2018.

[27] Gaurav Dugar, Alexander Herbig, Konrad Förstner, Nadja Heidrich, Richard Reinhardt, Kay Nieselt, and Cynthia M Sharma. High-resolution transcriptome maps reveal strain-specific regulatory features of multiple *Campylobacter jejuni* isolates. *PLoS genetics*, 9(5), 2013.

[28] Isabelle Rosinski-Chupin, Elisabeth Sauvage, Agnès Fouet, Claire Poyart, and Philippe Glaser. Conserved and specific features of *Streptococcus pyogenes* and *Streptococcus agalactiae* transcriptional landscapes. *BMC genomics*, 20(1):236, 2019.

[29] Carsten Kröger, Shane C Dillon, Andrew D S Cameron, Kai Papenfort, Sathesh K Sivasankaran, Karsten Hokamp, Yanjie Chao, Alexandra Sittka, Magali Hébrard, and Kristian Händler. The transcriptional landscape and small RNAs of *Salmonella enterica serovar Typhimurium*. *Proceedings of the National Academy of Sciences*, 109(20):1277–1286, 2012.

[30] Marco Albrecht, Cynthia M Sharma, Marcus T Dittrich, Tobias Müller, Richard Reinhardt, Jörg Vogel, and Thomas Rudel. The transcriptional landscape of *Chlamydia pneumoniae*. *Genome biology*, 12(10):98, 2011.

[31] Wenjun Shao, Morgan N Price, Adam M Deutschbauer, Margaret F Romine, and Adam P Arkin. Conservation of transcription start sites within genes across a bacterial genus. *MBio*, 5(4):e01398–14, 2014.

[32] Anna Zhukova, Luis Guilherme Fernandes, Perrine Hugon, Christopher J Pappas, Odile Sismeiro, Jean-Yves Coppée, Christophe Becavin, Christophe Malabat, Azad Eshghi, and Jun-Jie Zhang. Genome-wide transcriptional start site mapping and sRNA identification in the pathogen *Leptospira interrogans*. *Frontiers in cellular and infection microbiology*, 7:10, 2017.

[33] Yujin Jeong, Ji-Nu Kim, Min Woo Kim, Giselda Bucca, Suhyung Cho, Yeo Joon Yoon, Byung-Gee Kim, Jung-Hye Roe, Sun Chang Kim, and Colin P Smith. The dynamic transcriptional and translational landscape of the model antibiotic producer *Streptomyces coelicolor A3 (2)*. *Nature communications*, 7:11605, 2016.

[34] M Carla Martini, Ying Zhou, Huaming Sun, and Scarlet Sara Shell. Defining the transcriptional and post-transcriptional landscapes of *Mycobacterium smegmatis* in aerobic growth and hypoxia. *Frontiers in microbiology*, 10:591, 2019.

[35] Magali Boutard, Laurence Ettwiller, Tristan Cerisy, Adriana Alberti, Karine Labadie, Marcel Salanoubat, Ira Schildkraut, and Andrew C Tolonen. Global repositioning of transcription start sites in a plant-fermenting bacterium. *Nature communications*, 7(1):1–9, 2016.

[36] Yuling Liao, Lianggang Huang, Bin Wang, Feng Zhou, and Li Pan. The global transcriptional landscape of *Bacillus amyloliquefaciens XH7* and high-throughput screening of strong promoters based on RNA-seq data. *Gene*, 571(2):252–262, 2015.

[37] Paolo Di Tommaso, Maria Chatzou, Evan W Floden, Pablo Prieto Barja, Emilio Palumbo, and Cedric Notredame. Nextflow enables reproducible computational workflows. *Nature biotechnology*, 35(4):316–319, 2017.

[38] Aaron R Quinlan and Ira M Hall. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6):841–842, 2010.

[39] Genomic Paris Centre. genomicpariscentre/bedtools. `https://hub.docker.com/r/genomicpariscentre/bedtools/`, 2020.

[40] Cha Zhang and Yunqian Ma. *Ensemble machine learning: methods and applications*. Springer, 2012.

[41] Jae Won Lee, Jung Bok Lee, Mira Park, and Seuck Heun Song. An extensive comparison of recent classification tools applied to microarray data. *Computational Statistics & Data Analysis*, 48(4):869–885, 2005.

[42] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. KNN model-based approach in classification. In *OTM Confederated International Conferences*.

[43] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[44] Sotiris B Kotsiantis, George E Tsekouras, and Panayiotis E Pintelas. Bagging model trees for classification problems. In *Panhellenic Conference on Informatics*, pages 328–337, 2005.

[45] Leo Breiman. Arcing classifier (with discussion and a rejoinder by the author). *The annals of statistics*, 26(3):801–849, 1998.

[46] Grant Izmirlian. Application of the random forest classification algorithm to a SELDI-TOF proteomics study in the setting of a cancer prevention trial. *Annals of the New York Academy of Sciences*, 1020(1):154–174, 2004.

[47] Marc Kirchner, Wiebke Timm, Peying Fong, Philine Wangemann, and Hanno Steen. Non-linear classification for on-the-fly fractional mass filtering and targeted precursor fragmentation in mass spectrometry experiments. *Bioinformatics*, 26(6):791–797, 2010.

[48] Mark R Segal. Machine learning benchmarks and random forest regression. 2004.

[49] Stephen E Hamby and Jonathan D Hirst. Prediction of glycosylation sites using random forests. *BMC bioinformatics*, 9(1):500, 2008.

[50] Steven T Hill, Rachael Kuintzle, Amy Teegarden, Erich Merrill III, Padideh Danaee, and David A Hendrix. A deep recurrent neural network discovers complex biological rules to decipher RNA protein-coding potential. *Nucleic acids research*, 46(16):8105–8113, 2018.

[51] Zhen Shen, Wenzheng Bao, and De-Shuang Huang. Recurrent neural network for predicting transcription factor binding sites. *Scientific reports*, 8(1):1–10, 2018.

[52] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[53] Rob Knight, Gavin Huttley, and Daniel McDonald. scikit-bio. `http://scikit-bio.org/`, 2014.

[54] François Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[55] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[56] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[57] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[58] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, and Others. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[59] Susan Baldwin. Compute canada: advancing computational research. In *Journal of Physics: Conference Series*, volume 341, page 012001. IOP Publishing, 2012.

[60] Gael Varoquaux. Joblib: running Python functions as pipeline jobs. `https://github.com/joblib/joblib`, 2015.

[61] Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. The numpy array: a structure for efficient numerical computation. *CoRR*, abs/1102.1523, 2011.

[62] André Altmann, Laura Toloşi, Oliver Sander, and Thomas Lengauer. Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347, 04 2010.

[63] David Pribnow. Nucleotide sequence of an RNA polymerase binding site at an early T7 promoter. *Proceedings of the National Academy of Sciences*, 72(3):784–788, 1975.

[64] Heinz Schaller, Christopher Gray, and Karin Herrmann. Nucleotide sequence of an RNA polymerase binding site from the DNA of bacteriophage fd. *Proceedings of the National Academy of Sciences*, 72(2):737–741, 1975.

[65] Timothy L Bailey, Nadya Williams, Chris Misleh, and Wilfred W Li. MEME: discovering and analyzing DNA and protein sequence motifs. *Nucleic acids research*, 34(suppl_2):W369–W373, 2006.

[66] Ruben Chevez-Guardado and Lourdes Peña-Castillo. Promotech: A universal tool for promoter detection in bacterial genomes. `https://github.com/BioinformaticsLabAtMUN/PromoTech`, 2020.